IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventors:	Ronald S. Karr; Ramana Jonnala; Narasimha R. Valiveti; Dhanesh Joshi		
Assignee:	Symantec Operating Corporation		
Title:	TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE		
Application No.:	10/788,589	Filing Date:	February 27, 2004
Examiner:	Yaima Campos	Group Art Unit:	2185
Docket No.:	VRT0120US	Confirmation No.:	6846

Austin, Texas
September 11, 2009

MAIL STOP AMENDMENT
COMMISSIONER FOR PATENTS
P. O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF PRIOR INVENTION IN THE UNITED STATES PURSUANT TO 37 C.F.R. § 1.131

Dear Sir:

PURPOSE OF THE DECLARATION

The Office Action dated May 12, 2009 rejected Claims 1-2, 5-8, 11-16, 26-28 and 31-34 under 35 U.S.C. § 103(a) as being unpatentable by U.S. Patent No. 7,216,133 issued to Wu et al. ("Wu") in view of U.S. Patent Application Publication 2005/0055523 issued to Suishu et al. ("Suishu"). Also, Claims 9-10 and 35-36 stand rejected under 35 U.S.C. § 103(a) as purportedly being unpatentable by Wu in view of Suishu as applied to Claims 1 and 7 above, and further in view of U.S. Patent 6,088,697 issued to Crockett et al. ("Crockett").

DECLARATION

We declare as follows:

1. We are joint inventors of the claimed subject matter of the above-referenced application.

2. We conceived of the claimed subject matter of the above-referenced application in the United States prior to February 23, 2004, the effective date of Suishu.

3. Attached is Exhibit A. Exhibit A includes a hard copy of the first draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed.

4. Attached is Exhibit B. Exhibit B includes a hard copy of an e-mail sent to us on Friday, February 20, 2004, at 9:53 AM. The e-mail included as an attachment, the first draft of Exhibit A for our review.

5. Attached is Exhibit C. Exhibit C includes a hard copy of an e-mail sent by inventor Ronald S. Karr on Friday, February 20, 2004 at 8:17 PM including comments concerning the first draft of Exhibit A.

6. February 21, 2004 and February 22, 2004 fell on a Saturday and Sunday, respectively.

7. Attached is Exhibit D. Exhibit D includes a hard copy of an e-mail sent to us on Monday, February 23, 2004 at 5:37 AM.

8. Attached is Exhibit E. Exhibit E includes hard copy of a second draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed.

9. Attached is Exhibit F. Exhibit F includes a hard copy of an e-mail sent to us on Monday, February 23, 2004 at 6:56 AM. The e-mail included as an attachment the second draft of Exhibit E for our review.

10. Attached is Exhibit G. Exhibit G includes a hard copy of an e-mail sent by inventor Narasimha Valiveti on Monday, February 23, 2004 at 9:24 PM including comments concerning the second draft of Exhibit E.

11. Attached is Exhibit H. Exhibit H includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Tuesday, February 24, 2004 at 1:12 AM including comments concerning the second draft of Exhibit E.

12. Attached is Exhibit I. Exhibit I includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Wednesday, February 25, 2004 at 2:25 PM including comments concerning the second draft of Exhibit E.

13. Attached is Exhibit J. Exhibit J includes a hard copy of an e-mail sent by inventor Narasimha Valiveti on Wednesday, February 25, 2004 at 3:05 PM including comments concerning the second draft of Exhibit E.

14. Attached is Exhibit K. Exhibit K includes a hard copy of a final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit K also includes a hard copy of a declaration for patent application/power of attorney.

15. Attached is Exhibit L. Exhibit L includes a hard copy of an e-mail sent to us on Wednesday, February 25, 2004 at 5:38 PM. The e-mail included as an attachment the final draft of Exhibit K as well as a hard copy of the declaration for patent application/power of attorney of Exhibit L for our review.

16. Attached is Exhibit M. Exhibit M includes a hard copy of an e-mail sent to us on Wednesday, February 25, 2004 at 6:02 PM.

17. Attached is Exhibit N. Exhibit N includes a hard copy of an e-mail sent to us on Thursday, February 26, 2006 at 11:09 AM.

18. Attached is Exhibit O. Exhibit O includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Thursday, February 26, 2004 at 2:22 PM including comments on the final draft of Exhibit K.

19. Attached is Exhibit P. Exhibit P includes a hard copy of a first revised final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit P also includes a hard copy of a declaration for patent application/power of attorney.

20. Attached is Exhibit Q. Exhibit Q includes a hard copy of an e-mail sent to us on Thursday, February 26, 2004 at 3:37 PM. The e-mail included as an attachment the first revised final draft of Exhibit P as well as a hard copy of the declaration for patent application/power of attorney of Exhibit P for our review.

21. Attached is Exhibit R. Exhibit R includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Thursday, February 26, 2004 at 4:35 PM, including comments on the first revised final draft of Exhibit P.

22. Attached is Exhibit S. Exhibit S includes a hard copy of a second revised final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit S also includes a hard copy of a declaration for patent application/power of attorney.

23. Attached is Exhibit T. Exhibit T includes a hard copy of an e-mail sent to us on Thursday, February 26, 2004 at 5:53 PM. The e-mail included as an attachment the second revised final draft of Exhibit S as well as a hard copy of the declaration for patent application/power of attorney of Exhibit S for our review.

24. Attached is Exhibit U. Exhibit U includes a hard copy of a third revised final draft of the above-referenced application, which is identical to the above-referenced application as filed. Exhibit U also includes a hard copy of a declaration for patent application/power of attorney.

25. Attached is Exhibit V. Exhibit V includes a hard copy of an e-mail sent to us on Friday, February 27, 2004 at 10:45 AM. The e-mail included as an attachment the third revised final draft of Exhibit U as well as a hard copy of the declaration for patent application/power of attorney of Exhibit U for our review.

26. Exhibits A-V establish conception of the invention prior to the effective date of Suishu (February 23, 2004) coupled with due diligence from prior to said data to a subsequent reduction of practice or to the filing of the application (February 27, 2004).

27. The inventive concepts shown in Exhibits A, E, K, P, S, and U were conceived by Ronald S. Karr; Ramana Jonnala; Narasimha R. Valiveti; Dhanesh Joshi.

28. All of the enumerated acts took place in the United States of America.

****

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under §1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the Application or any patent issued thereon.

Full Name of Inventor: Ronald S. Karr

Inventor's Signature: ______________________

Date: 10/8/2004

Country of Citizenship: U.S.A.

Residence: 333 Ramona Street
Palo Alto, California 94301

Full Name of Inventor: Ramana Jonnala

Inventor's Signature: ______________________________

Date: ______________________________

Country of Citizenship: U.S.A.

Residence: 825 E. Evelyn Avenue, #326
Sunnyvale, California 94086

Full Name of Inventor: Narasimha R. Valiveti

Inventor's Signature: ______________________________

Date: ______________________________

Country of Citizenship: India

Residence: 395 Ano Nuevo Avenue, Apt. #413
Sunnyvale, California 94085

Full Name of Inventor: Dhanesh Joshi

Inventor's Signature: ______________________________

Date: ______________________________

Country of Citizenship: India

Residence: 46 Cabot Avenue
Santa Clara, California 95051

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventors:	Ronald S. Karr; Ramana Jonnala; Narasimha R. Valiveti; Dhanesh Joshi		
Assignee:	Symantec Operating Corporation		
Title:	TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE		
Application No.:	10/788,589	Filing Date:	February 27, 2004
Examiner:	Yaima Campos	Group Art Unit:	2185
Docket No.:	VRT0120US	Confirmation No.:	6846

Austin, Texas
September 11, 2009

MAIL STOP AMENDMENT
COMMISSIONER FOR PATENTS
P. O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF PRIOR INVENTION IN THE UNITED STATES PURSUANT TO 37 C.F.R. § 1.131

Dear Sir:

PURPOSE OF THE DECLARATION

The Office Action dated May 12, 2009 rejected Claims 1-2, 5-8, 11-16, 26-28 and 31-34 under 35 U.S.C. § 103(a) as being unpatentable by U.S. Patent No. 7,216,133 issued to Wu et al. ("Wu") in view of U.S. Patent Application Publication 2005/0055523 issued to Suishu et al. ("Suishu"). Also, Claims 9-10 and 35-36 stand rejected under 35 U.S.C. § 103(a) as purportedly being unpatentable by Wu in view of Suishu as applied to Claims 1 and 7 above, and further in view of U.S. Patent 6,088,697 issued to Crockett et al. ("Crockett").

DECLARATION

We declare as follows:

1. We are joint inventors of the claimed subject matter of the above-referenced application.

2. We conceived of the claimed subject matter of the above-referenced application in the United States prior to February 23, 2004, the effective date of Suishu.

3. Attached is Exhibit A. Exhibit A includes a hard copy of the first draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed.

4. Attached is Exhibit B. Exhibit B includes a hard copy of an e-mail sent to us on Friday, February 20, 2004, at 9:53 AM. The e-mail included as an attachment, the first draft of Exhibit A for our review.

5. Attached is Exhibit C. Exhibit C includes a hard copy of an e-mail sent by inventor Ronald S. Karr on Friday, February 20, 2004 at 8:17 PM including comments concerning the first draft of Exhibit A.

6. February 21, 2004 and February 22, 2004 fell on a Saturday and Sunday, respectively.

7. Attached is Exhibit D. Exhibit D includes a hard copy of an e-mail sent to us on Monday, February 23, 2004 at 5:37 AM.

8. Attached is Exhibit E. Exhibit E includes hard copy of a second draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed.

9. Attached is Exhibit F. Exhibit F includes a hard copy of an e-mail sent to us on Monday, February 23, 2004 at 6:56 AM. The e-mail included as an attachment the second draft of Exhibit E for our review.

10. Attached is Exhibit G. Exhibit G includes a hard copy of an e-mail sent by inventor Narasimha Valiveti on Monday, February 23, 2004 at 9:24 PM including comments concerning the second draft of Exhibit E.

11. Attached is Exhibit H. Exhibit H includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Tuesday, February 24, 2004 at 1:12 AM including comments concerning the second draft of Exhibit E.

12. Attached is Exhibit I. Exhibit I includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Wednesday, February 25, 2004 at 2:25 PM including comments concerning the second draft of Exhibit E.

13. Attached is Exhibit J. Exhibit J includes a hard copy of an e-mail sent by inventor Narasimha Valiveti on Wednesday, February 25, 2004 at 3:05 PM including comments concerning the second draft of Exhibit E.

14. Attached is Exhibit K. Exhibit K includes a hard copy of a final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit K also includes a hard copy of a declaration for patent application/power of attorney.

15. Attached is Exhibit L. Exhibit L includes a hard copy of an e-mail sent to us on Wednesday, February 25, 2004 at 5:38 PM. The e-mail included as an attachment the final draft of Exhibit K as well as a hard copy of the declaration for patent application/power of attorney of Exhibit L for our review.

16. Attached is Exhibit M. Exhibit M includes a hard copy of an e-mail sent to us on Wednesday, February 25, 2004 at 6:02 PM.

17. Attached is Exhibit N. Exhibit N includes a hard copy of an e-mail sent to us on Thursday, February 26, 2006 at 11:09 AM.

18. Attached is Exhibit O. Exhibit O includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Thursday, February 26, 2004 at 2:22 PM including comments on the final draft of Exhibit K.

19. Attached is Exhibit P. Exhibit P includes a hard copy of a first revised final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit P also includes a hard copy of a declaration for patent application/power of attorney.

20. Attached is Exhibit Q. Exhibit Q includes a hard copy of an e-mail sent to us on Thursday, February 26, 2004 at 3:37 PM. The e-mail included as an attachment the first revised final draft of Exhibit P as well as a hard copy of the declaration for patent application/power of attorney of Exhibit P for our review.

21. Attached is Exhibit R. Exhibit R includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Thursday, February 26, 2004 at 4:35 PM, including comments on the first revised final draft of Exhibit P.

22. Attached is Exhibit S. Exhibit S includes a hard copy of a second revised final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit S also includes a hard copy of a declaration for patent application/power of attorney.

23. Attached is Exhibit T. Exhibit T includes a hard copy of an e-mail sent to us on Thursday, February 26, 2004 at 5:53 PM. The e-mail included as an attachment the second revised final draft of Exhibit S as well as a hard copy of the declaration for patent application/power of attorney of Exhibit S for our review.

24. Attached is Exhibit U. Exhibit U includes a hard copy of a third revised final draft of the above-referenced application, which is identical to the above-referenced application as filed. Exhibit U also includes a hard copy of a declaration for patent application/power of attorney.

25. Attached is Exhibit V. Exhibit V includes a hard copy of an e-mail sent to us on Friday, February 27, 2004 at 10:45 AM. The e-mail included as an attachment the third revised final draft of Exhibit U as well as a hard copy of the declaration for patent application/power of attorney of Exhibit U for our review.

26. Exhibits A-V establish conception of the invention prior to the effective date of Suishu (February 23, 2004) coupled with due diligence from prior to said data to a subsequent reduction of practice or to the filing of the application (February 27, 2004).

27. The inventive concepts shown in Exhibits A, E, K, P, S, and U were conceived by Ronald S. Karr; Ramana Jonnala; Narasimha R. Valiveti; Dhanesh Joshi.

28. All of the enumerated acts took place in the United States of America.

****

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under §1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the Application or any patent issued thereon.

Full Name of Inventor: Ronald S. Karr

Inventor's Signature: ______________________________

Date: ______________________________

Country of Citizenship: U.S.A.

Residence: 333 Ramona Street
Palo Alto, California 94301

Full Name of Inventor: Ramana Jonnala

Inventor's Signature: ______________________

Date: 10/14/09

Country of Citizenship: U.S.A.

Residence: 825 E. Evelyn Avenue, #326
Sunnyvale, California 94086

Full Name of Inventor: Narasimha R. Valiveti

Inventor's Signature: ______________________

Date: ______________________

Country of Citizenship: India

Residence: 395 Ano Nuevo Avenue, Apt. #413
Sunnyvale, California 94085

Full Name of Inventor: Dhanesh Joshi

Inventor's Signature: ______________________

Date: ______________________

Country of Citizenship: India

Residence: 46 Cabot Avenue
Santa Clara, California 95051

PATENT

In the United States Patent and Trademark Office

Inventors:	Ronald S. Karr; Ramana Jonnala; Narasimha R. Valiveti; Dhanesh Joshi		
Assignee:	Symantec Operating Corporation		
Title:	TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE		
Application No.:	10/788,589	Filing Date:	February 27, 2004
Examiner:	Yaima Campos	Group Art Unit:	2185
Docket No.:	VRT0120US	Confirmation No.:	6846

Austin, Texas
September 11, 2009

MAIL STOP AMENDMENT
COMMISSIONER FOR PATENTS
P. O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF PRIOR INVENTION IN THE UNITED STATES PURSUANT TO 37 C.F.R. § 1.131

Dear Sir:

PURPOSE OF THE DECLARATION

The Office Action dated May 12, 2009 rejected Claims 1-2, 5-8, 11-16, 26-28 and 31-34 under 35 U.S.C. § 103(a) as being unpatentable by U.S. Patent No. 7,216,133 issued to Wu et al. ("Wu") in view of U.S. Patent Application Publication 2005/0055523 issued to Suishu et al. ("Suishu"). Also, Claims 9-10 and 35-36 stand rejected under 35 U.S.C. § 103(a) as purportedly being unpatentable by Wu in view of Suishu as applied to Claims 1 and 7 above, and further in view of U.S. Patent 6,088,697 issued to Crockett et al. ("Crockett").

DECLARATION

We declare as follows:

1. We are joint inventors of the claimed subject matter of the above-referenced application.

2. We conceived of the claimed subject matter of the above-referenced application in the United States prior to February 23, 2004, the effective date of Suishu.

3. Attached is Exhibit A. Exhibit A includes a hard copy of the first draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed.

4. Attached is Exhibit B. Exhibit B includes a hard copy of an e-mail sent to us on Friday, February 20, 2004, at 9:53 AM. The e-mail included as an attachment, the first draft of Exhibit A for our review.

5. Attached is Exhibit C. Exhibit C includes a hard copy of an e-mail sent by inventor Ronald S. Karr on Friday, February 20, 2004 at 8:17 PM including comments concerning the first draft of Exhibit A.

6. February 21, 2004 and February 22, 2004 fell on a Saturday and Sunday, respectively.

7. Attached is Exhibit D. Exhibit D includes a hard copy of an e-mail sent to us on Monday, February 23, 2004 at 5:37 AM.

8. Attached is Exhibit E. Exhibit E includes hard copy of a second draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed.

9. Attached is Exhibit F. Exhibit F includes a hard copy of an e-mail sent to us on Monday, February 23, 2004 at 6:56 AM. The e-mail included as an attachment the second draft of Exhibit E for our review.

10. Attached is Exhibit G. Exhibit G includes a hard copy of an e-mail sent by inventor Narasimha Valiveti on Monday, February 23, 2004 at 9:24 PM including comments concerning the second draft of Exhibit E.

11. Attached is Exhibit H. Exhibit H includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Tuesday, February 24, 2004 at 1:12 AM including comments concerning the second draft of Exhibit E.

12. Attached is Exhibit I. Exhibit I includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Wednesday, February 25, 2004 at 2:25 PM including comments concerning the second draft of Exhibit E.

13. Attached is Exhibit J. Exhibit J includes a hard copy of an e-mail sent by inventor Narasimha Valiveti on Wednesday, February 25, 2004 at 3:05 PM including comments concerning the second draft of Exhibit E.

14. Attached is Exhibit K. Exhibit K includes a hard copy of a final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit K also includes a hard copy of a declaration for patent application/power of attorney.

15. Attached is Exhibit L. Exhibit L includes a hard copy of an e-mail sent to us on Wednesday, February 25, 2004 at 5:38 PM. The e-mail included as an attachment the final draft of Exhibit K as well as a hard copy of the declaration for patent application/power of attorney of Exhibit L for our review.

16. Attached is Exhibit M. Exhibit M includes a hard copy of an e-mail sent to us on Wednesday, February 25, 2004 at 6:02 PM.

17. Attached is Exhibit N. Exhibit N includes a hard copy of an e-mail sent to us on Thursday, February 26, 2006 at 11:09 AM.

18. Attached is Exhibit O. Exhibit O includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Thursday, February 26, 2004 at 2:22 PM including comments on the final draft of Exhibit K.

19. Attached is Exhibit P. Exhibit P includes a hard copy of a first revised final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit P also includes a hard copy of a declaration for patent application/power of attorney.

20. Attached is Exhibit Q. Exhibit Q includes a hard copy of an e-mail sent to us on Thursday, February 26, 2004 at 3:37 PM. The e-mail included as an attachment the first revised final draft of Exhibit P as well as a hard copy of the declaration for patent application/power of attorney of Exhibit P for our review.

21. Attached is Exhibit R. Exhibit R includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Thursday, February 26, 2004 at 4:35 PM, including comments on the first revised final draft of Exhibit P.

22. Attached is Exhibit S. Exhibit S includes a hard copy of a second revised final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit S also includes a hard copy of a declaration for patent application/power of attorney.

23. Attached is Exhibit T. Exhibit T includes a hard copy of an e-mail sent to us on Thursday, February 26, 2004 at 5:53 PM. The e-mail included as an attachment the second revised final draft of Exhibit S as well as a hard copy of the declaration for patent application/power of attorney of Exhibit S for our review.

24. Attached is Exhibit U. Exhibit U includes a hard copy of a third revised final draft of the above-referenced application, which is identical to the above-referenced application as filed. Exhibit U also includes a hard copy of a declaration for patent application/power of attorney.

25. Attached is Exhibit V. Exhibit V includes a hard copy of an e-mail sent to us on Friday, February 27, 2004 at 10:45 AM. The e-mail included as an attachment the third revised final draft of Exhibit U as well as a hard copy of the declaration for patent application/power of attorney of Exhibit U for our review.

26. Exhibits A-V establish conception of the invention prior to the effective date of Suishu (February 23, 2004) coupled with due diligence from prior to said data to a subsequent reduction of practice or to the filing of the application (February 27, 2004).

27. The inventive concepts shown in Exhibits A, E, K, P, S, and U were conceived by Ronald S. Karr; Ramana Jonnala; Narasimha R. Valiveti; Dhanesh Joshi.

28. All of the enumerated acts took place in the United States of America.

****

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under §1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the Application or any patent issued thereon.

Full Name of Inventor: Ronald S. Karr

Inventor's Signature: ______________________________

Date: ______________________________

Country of Citizenship: U.S.A.

Residence: 333 Ramona Street
Palo Alto, California 94301

Full Name of Inventor: Ramana Jonnala

Inventor's Signature: ______________________

Date: ______________________

Country of Citizenship: U.S.A.

Residence: 825 E. Evelyn Avenue, #326
Sunnyvale, California 94086

Full Name of Inventor: Narasimha R. Valiveti

Inventor's Signature: V. Narasimha Rao

Date: 09/24/2009.

Country of Citizenship: India

Residence: ~~395 Ano Nuevo Avenue, Apt. #413~~ VNR
VNR ~~Sunnyvale, California 94085~~
3174 ADAMSWOOD DR
SAN JOSE, CA 95148

Full Name of Inventor: Dhanesh Joshi

Inventor's Signature: ______________________

Date: ______________________

Country of Citizenship: India

Residence: 46 Cabot Avenue
Santa Clara, California 95051

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventors:	Ronald S. Karr; Ramana Jonnala; Narasimha R. Valiveti; Dhanesh Joshi		
Assignee:	Symantec Operating Corporation		
Title:	TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE		
Application No.:	10/788,589	Filing Date:	February 27, 2004
Examiner:	Yaima Campos	Group Art Unit:	2185
Docket No.:	VRT0120US	Confirmation No.:	6846

Austin, Texas
September 11, 2009

MAIL STOP AMENDMENT
COMMISSIONER FOR PATENTS
P. O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF PRIOR INVENTION IN THE UNITED STATES PURSUANT TO 37 C.F.R. § 1.131

Dear Sir:

PURPOSE OF THE DECLARATION

The Office Action dated May 12, 2009 rejected Claims 1-2, 5-8, 11-16, 26-28 and 31-34 under 35 U.S.C. § 103(a) as being unpatentable by U.S. Patent No. 7,216,133 issued to Wu et al. ("Wu") in view of U.S. Patent Application Publication 2005/0055523 issued to Suishu et al. ("Suishu"). Also, Claims 9-10 and 35-36 stand rejected under 35 U.S.C. § 103(a) as purportedly being unpatentable by Wu in view of Suishu as applied to Claims 1 and 7 above, and further in view of U.S. Patent 6,088,697 issued to Crockett et al. ("Crockett").

DECLARATION

We declare as follows:

1. We are joint inventors of the claimed subject matter of the above-referenced application.

2. We conceived of the claimed subject matter of the above-referenced application in the United States prior to February 23, 2004, the effective date of Suishu.

3. Attached is Exhibit A. Exhibit A includes a hard copy of the first draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed.

4. Attached is Exhibit B. Exhibit B includes a hard copy of an e-mail sent to us on Friday, February 20, 2004, at 9:53 AM. The e-mail included as an attachment, the first draft of Exhibit A for our review.

5. Attached is Exhibit C. Exhibit C includes a hard copy of an e-mail sent by inventor Ronald S. Karr on Friday, February 20, 2004 at 8:17 PM including comments concerning the first draft of Exhibit A.

6. February 21, 2004 and February 22, 2004 fell on a Saturday and Sunday, respectively.

7. Attached is Exhibit D. Exhibit D includes a hard copy of an e-mail sent to us on Monday, February 23, 2004 at 5:37 AM.

8. Attached is Exhibit E. Exhibit E includes hard copy of a second draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed.

9. Attached is Exhibit F. Exhibit F includes a hard copy of an e-mail sent to us on Monday, February 23, 2004 at 6:56 AM. The e-mail included as an attachment the second draft of Exhibit E for our review.

10. Attached is Exhibit G. Exhibit G includes a hard copy of an e-mail sent by inventor Narasimha Valiveti on Monday, February 23, 2004 at 9:24 PM including comments concerning the second draft of Exhibit E.

11. Attached is Exhibit H. Exhibit H includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Tuesday, February 24, 2004 at 1:12 AM including comments concerning the second draft of Exhibit E.

12. Attached is Exhibit I. Exhibit I includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Wednesday, February 25, 2004 at 2:25 PM including comments concerning the second draft of Exhibit E.

13. Attached is Exhibit J. Exhibit J includes a hard copy of an e-mail sent by inventor Narasimha Valiveti on Wednesday, February 25, 2004 at 3:05 PM including comments concerning the second draft of Exhibit E.

14. Attached is Exhibit K. Exhibit K includes a hard copy of a final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit K also includes a hard copy of a declaration for patent application/power of attorney.

15. Attached is Exhibit L. Exhibit L includes a hard copy of an e-mail sent to us on Wednesday, February 25, 2004 at 5:38 PM. The e-mail included as an attachment the final draft of Exhibit K as well as a hard copy of the declaration for patent application/power of attorney of Exhibit L for our review.

16. Attached is Exhibit M. Exhibit M includes a hard copy of an e-mail sent to us on Wednesday, February 25, 2004 at 6:02 PM.

17. Attached is Exhibit N. Exhibit N includes a hard copy of an e-mail sent to us on Thursday, February 26, 2006 at 11:09 AM.

18. Attached is Exhibit O. Exhibit O includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Thursday, February 26, 2004 at 2:22 PM including comments on the final draft of Exhibit K.

19. Attached is Exhibit P. Exhibit P includes a hard copy of a first revised final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit P also includes a hard copy of a declaration for patent application/power of attorney.

20. Attached is Exhibit Q. Exhibit Q includes a hard copy of an e-mail sent to us on Thursday, February 26, 2004 at 3:37 PM. The e-mail included as an attachment the first revised final draft of Exhibit P as well as a hard copy of the declaration for patent application/power of attorney of Exhibit P for our review.

21. Attached is Exhibit R. Exhibit R includes a hard copy of an e-mail sent by inventor Dhanesh Joshi on Thursday, February 26, 2004 at 4:35 PM, including comments on the first revised final draft of Exhibit P.

22. Attached is Exhibit S. Exhibit S includes a hard copy of a second revised final draft of the above-referenced application, which includes substantially similar subject matter as the above-referenced application as filed. Exhibit S also includes a hard copy of a declaration for patent application/power of attorney.

23. Attached is Exhibit T. Exhibit T includes a hard copy of an e-mail sent to us on Thursday, February 26, 2004 at 5:53 PM. The e-mail included as an attachment the second revised final draft of Exhibit S as well as a hard copy of the declaration for patent application/power of attorney of Exhibit S for our review.

24. Attached is Exhibit U. Exhibit U includes a hard copy of a third revised final draft of the above-referenced application, which is identical to the above-referenced application as filed. Exhibit U also includes a hard copy of a declaration for patent application/power of attorney.

25. Attached is Exhibit V. Exhibit V includes a hard copy of an e-mail sent to us on Friday, February 27, 2004 at 10:45 AM. The e-mail included as an attachment the third revised final draft of Exhibit U as well as a hard copy of the declaration for patent application/power of attorney of Exhibit U for our review.

26. Exhibits A-V establish conception of the invention prior to the effective date of Suishu (February 23, 2004) coupled with due diligence from prior to said data to a subsequent reduction of practice or to the filing of the application (February 27, 2004).

27. The inventive concepts shown in Exhibits A, E, K, P, S, and U were conceived by Ronald S. Karr; Ramana Jonnala; Narasimha R. Valiveti; Dhanesh Joshi.

28. All of the enumerated acts took place in the United States of America.

****

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under §1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the Application or any patent issued thereon.

Full Name of Inventor: Ronald S. Karr

Inventor's Signature: ______________________

Date: ______________________

Country of Citizenship: U.S.A.

Residence: 333 Ramona Street
Palo Alto, California 94301

Full Name of Inventor: Ramana Jonnala

Inventor's Signature: ______________________

Date: ______________________

Country of Citizenship: U.S.A.

Residence: 825 E. Evelyn Avenue, #326
Sunnyvale, California 94086

Full Name of Inventor: Narasimha R. Valiveti

Inventor's Signature: ______________________

Date: ______________________

Country of Citizenship: India

Residence: 395 Ano Nuevo Avenue, Apt. #413
Sunnyvale, California 94085

Full Name of Inventor: Dhanesh Joshi

Inventor's Signature: ______________________

Date: 10/12/2009

Country of Citizenship: India

Residence: 46 Cabot Avenue
Santa Clara, California 95051

EXHIBIT A

ATTORNEY/CLIENT PRIVILEGED AND CONFIDENTIAL

THIS DRAFT PATENT APPLICATION CONTAINS INFORMATION CONFIDENTIAL TO *VERITAS Software Corporation* AND INCLUDES PRIVILEGED, ATTORNEY-CLIENT COMMUNICATIONS

EXPRESS MAIL LABEL NO.:

(EV 304738055 US)

DRAFT

8/10/2009; 3:02:48 PM

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha Valiveti
Dhanesh Joshi

BACKGROUND OF THE INVENTION

[0001] Large scale data processing systems typically include several data storage subsystems (e.g., disk arrays) each containing many physical storage devices (e.g., hard disks) for storing critical data. Data processing systems often employ storage management systems to aggregate these physical storage devices to create highly reliable data storage. There are many types of highly reliable storage. Mirrored volume storage is an example. Mirrored volume storage replicates data over two or more mirrors of equal size. A logical memory block n of a mirrored volume maps to the same logical memory block n of each mirror. In turn, each logical memory block n of the mirrors map directly or indirectly to one or more physical memory blocks of one or more physical devices. Mirrored volumes provide data redundancy. If an application is unable to access data of one, a duplicate of the data sought should be available in an alternate mirror.

[0002] Figure 1 illustrates relevant components of an exemplary data processing system 10 that employs a two-way mirrored volume. While the present invention will be described with reference to a two-way mirrored volume, the present invention should not be limited thereto. The present invention may find use with other types of redundant storage including, for example, a three-way mirrored storage volume. Data processing system 10 includes a host (e.g., server computer system) 12 coupled to data storage subsystems 14 and 16 via storage interconnect 20. For purposes of explanation, storage interconnect 20 will take form in a storage area network (SAN) it being understood that the term storage interconnect should not be limited thereto. SAN 20 may include devices (e.g., switches, routers, hubs, etc.) that cooperate to transmit input/output (IO) transactions between host 12 and storage subsystems 14 and 16.

[0003] Each of the data storage subsystems 14 and 16 includes several physical storage devices. For purposes of explanation, data storage subsystems 14 and 16 are assumed to include several hard disks. The term physical storage device should not be limited to hard disks. Data storage subsystems 14 and 16 may take different forms. For example, data storage subsystem 14 may consist of "Just a Bunch of Disks" (JBOD) connected to an array controller card. Data storage subsystem 16 may consist of a block server appliance. For purposes of explanation, each of the data storage subsystems 14 and 16 will take form in an intelligent disk array, it being understood that the term data storage subsystem should not be limited thereto.

[0004] As noted, each of the disk arrays 14 and 16 includes several hard disks. The hard disk is the most popular permanent storage device currently used. A hard disk's total storage capacity is divided into many small chunks called physical memory blocks. For example, a 10 GB hard disk contains 20 million physical memory blocks, with each block able to hold 512 bytes of data. Any random physical memory block can be written to or read from in about the same time, without first having to read or write other physical memory blocks. Once written, a physical memory block continues to hold data even after the hard disk is powered down. While hard disks in general are reliable, they are subject to occasional failure. Data systems employ data redundancy schemes such as mirrored volumes to protect against occasional failure of a hard disk.

[0005] Host 12 includes an application 22 executing on one or more processors. Application 22 generates IO transactions to access critical data in response to receiving requests from client computer systems (not shown) coupled to host 12. In addition to application 22, host 12 includes a storage manager 24 executing on one or more processors. Volume Manager^{TM} provided by VERITAS Software Corporation of Mountain View, California, is an exemplary storage manager. Although many of the examples described herein will emphasize virtualization architecture and terminology associated with the VERITAS Volume Manager™, the software and techniques described herein can be used with a variety of different storage managers and architectures.

[0006] Storage managers can perform several functions. More particularly, storage managers can create storage objects (also known as virtualized disks) by aggregating hard disks such as those of disk arrays 14 and 16, underlying storage objects, or both. A storage object is an abstraction. Figure 2 shows a visual representation of exemplary storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ created for use in data processing system 10. Each of the storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ in Figure 2 consists of an array of n_{max} logical memory blocks that store or are configured to store data. While it is said that a logical memory block stores or is configured to store data, in reality the data is stored in one or more physical memory blocks of hard disks allocated directly or indirectly to the logical memory block.

[0007] Storage objects aggregated from hard disks can themselves be aggregated to form storage objects called logical data volumes. Logical data volumes are typically presented for direct or indirect use by an application such as application 22 executing on host 12. Thus, application 22 generates IO transactions to read data from or write data to one or more logical memory blocks of a data volume not knowing that the data volume is an aggregation of underlying storage objects, which in turn are aggregations of hard disks. Properties of storage objects depend on how the underlying storage objects or hard disks are aggregated.

In other words, the method of aggregation determines the storage object type. In theory, there are a large number of possible methods of aggregation. The more common forms of aggregation include concatenated storage, striped storage, RAID storage, or mirrored storage. A more thorough discussion of how storage objects or hard disks can be aggregated can be found within Dilip M. Ranade [2002], "Shared Data Clusters" Wiley Publishing, Inc., which is incorporated herein by reference in its entirety.

[0008] Mirrored volumes provide highly reliable access to critical data. $V_{Example}$ of Figure 2 is an exemplary two-way mirrored volume. $V_{Example}$ was created by aggregating underlying storage objects (hereinafter mirrors) $M0_{Example}$ and $M1_{Example}$. Mirrors $M0_{Example}$ and $M1_{Example}$ were created by concatenating physical memory blocks from hard disks $d0_{Example}$ and $d1_{Example}$ (not shown) in disk arrays 14 and 16, respectively.

[0009] Storage managers can create storage object descriptions that describe the relationship between storage objects and their underlying storage objects or hard disks. These storage object descriptions typically include configuration maps. It is noted that storage object descriptions may include other information such as information indicating that a storage object is a snapshot copy of another storage object.

[0010] A configuration map maps a logical memory block of a corresponding storage object to one or more logical memory blocks of one or more underlying storage objects or to one or more physical memory blocks of one or more hard disks. To illustrate, configuration maps $CMV_{Example}$, $CMM0_{Example}$, and $CMM1_{Example}$ are created for mirrored volume $V_{Example}$ and underlying mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Configuration map $CMV_{Example}$ maps each logical memory block n of $V_{Example}$ to logical memory blocks n of mirrors $M0_{Example}$ and $M1_{Example}$. Configuration map $CMM0_{Example}$ maps each logical memory block n of mirror $M0_{Example}$ to a physical memory block x in hard disk $d0_{Example}$, while configuration map $CMM1_{Example}$ maps each logical memory block n of mirror $M1_{Example}$ to a physical memory block y in hard disk $d1_{Example}$. Configuration map $CMV_{Example}$ can be provided for use by storage manager 24, while configuration maps $CMM0_{Example}$ and $CMM1_{Example}$ can be provided for use by storage managers 34 and 36 executing on one or more processors in disk arrays 14 and 16, respectively.

[0011] Storage managers use configuration maps to translate IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks. To illustrate, presume an IO transaction is generated by application 22 to write data D to logical memory block 3 of data volume $V_{Example}$. This IO transaction is received directly or indirectly by storage manager 24. In turn, storage manager 24 accesses configuration map $CMV_{Example}$ and learns that logical memory block 3 is mapped to logical block 3 in both mirrors $M0_{Example}$ and $M1_{Example}$. It is noted storage manager may not receive the exact IO transaction generated by application 22. However, the transaction storage manager 24 receives will indicate that data D is to be written to block 3 of mirrored volume $V_{Example}$.

[0012] Storage manager 24 then generates first and second IO transactions to write data D to logical blocks 3 and mirrors $M0_{Example}$ and $M1_{Example}$, respectively. The IO transactions generated by storage manager 24 are transmitted to disk arrays 14 and 16, respectively, via SAN 20. Storage managers 34 and 36 of disk arrays 14 and 16, respectively receive directly or indirectly, the IO transactions sent by storage manager 24. It is noted that the IO transactions received by storage managers 34 and 36 may not be the exact IO transactions generated and sent by storage manager 24. Nonetheless, storage managers 34 and 36 will each receive an IO transaction to write data D to logical block 3 in mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Storage manager 34, in response to receiving the IO transaction, accesses configuration map $CMM0_{Example}$ to learn that block 3 of mirror $M0_{Example}$ is mapped to, for example, physical memory block 200 within hard disk d0. In response, a transaction is generated to write data D to physical memory block 200 within disk d0. Storage manager 36 accesses configuration map $CMM1_{Example}$ to learn that logical block 3 of mirror $M1_{Example}$ is mapped to, for example, physical memory block 300 within hard disk d1. In response, an IO transaction is generated for writing data D to physical memory block 300 within hard disk d1.

[0013] As noted above, while hard disks are reliable, hard disks are subject to failure. Data may be inaccessible within a failed hard disk. For this reason and others, administrators create mirrored data volumes. Unfortunately, data within mirrors of a mirrored volume may get into an inconsistent state if, for example, there is a server crash, storage power failure, or other problem which prevents data from being properly written to a hard disk. Consider the exemplary mirrored volume $V_{Example}$. Presume storage manager 24 generates first and second IO transactions to write data D to logical blocks 3 in mirrors $M0_{Example}$ and $M1_{Example}$ in response to the IO transaction generated by application 22. Further, presume host 12 may fail after the first IO transaction is transmitted to disk array 14, but before the second IO transaction is transmitted to disk array 16. As a result, data D is written to physical memory block 200 within disk d0 of disk array 14, but data D is not written to physical memory block 300 within hard disk d1. When host 12 is restarted and exemplary volume $V_{Example}$ is made available again to application 22, mirrors $MO_{Example}$ and $M1_{Example}$ are said to be out of sync. In other words, mirrors $M0_{Example}$ and $M1_{Example}$ are no longer identical since at least block 3 in each contain different data. An IO transaction to read from logical memory block 3 of mirrored volume $V_{Example}$ could return either old or new data depending on whether the data is read from physical memory block 200 of hard disk d0 or physical memory block 300 of hard disk d1. Mirrors $MO_{Example}$ and $M1_{Example}$ should be resychronized before either is accessed again.

[0014] A brute force method to resynchronize mirrors $M0_{Example}$ and $M1_{Example}$ is simply to presume that one mirror (e.g., $M0_{Example}$) contains correct data and copy the contents of the one mirror to the other (e.g., $M1_{Example}$). It can take hours to resynchronize using this method. A smarter resynchronization technique is possible, but it requires some preparation. This alternate technique involves using what is called a dirty region map. Figure 2 illustrates a visual representation of an exemplary dirty region map 40 consisting of n_{max} entries corresponding to the n_{max} logical memory blocks within mirrors $M0_{Example}$ and $M1_{Example}$. Each entry of the dirty region map 40 indicates whether the corresponding blocks in mirrors are considered synchronized. For example, if entry n is set to logical 1, then blocks n in the mirrors are considered out of synchronization, and if n is set to logical 0, blocks n in the mirrors are considered in synchronization. Entry n in dirty region map 40 is set to logical 1 when the application 22 generates a transaction for writing data to logical block n of volume

$V_{Example}$. Entry n is maintained in the logical 1 state until acknowledgement is received from both disk arrays 14 and 16 that data D has been written successfully to the physical memory blocks allocated to logical memory block n. However, if, using the example above, the first IO transaction to write data D to block 3 of mirror $M0_{Example}$ succeeds while the second IO transaction to write to block 3 of $M1_{Example}$ fails, then entry 3 and dirty region map 40 will be maintained as a logical 1 indicating that logical blocks 3 in the mirrors are out of synchronization. Mirrors $M0_{Example}$ and $M1_{Example}$ can be resynchronized by copying data from $M0_{Example}$ to mirror $M1_{Example}$, but for only logical memory blocks corresponding to dirty region map entries set to logical 1.

SUMMARY OF THE INVENTION

[0015]

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

Fig. 1 illustrates a block diagram of a data system in which a logical data volume may be created;

Fig. 2 illustrates a visual representation of the storage objects created in the data system of Fig. 1;

Fig. 3, illustrates a block diagram of a data system capable of implementing one embodiment of the present invention;

Figs. 4 shows visual representations of a mirrored data volume employed in the data system of Fig. 3;

Fig. 5 is a visual representation of exemplary tag tables employed in the disk arrays of Fig. 3;

Fig.6 is a flow chart illustrating relevant aspects of managing the tag tables shown in Fig. 5.

[0017] The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION

[0018] Figure 3 illustrates in block diagram form, relevant components of an exemplary data processing system 40 capable of employing one embodiment of the present invention. Data processing system 40 includes a host (e.g., server computer system) 42 coupled to data storage subsystems 44 an 46 via storage interconnect 50. The present invention should not be limited to use in a data processing system consisting of only two storage subsystems. The present invention may be employed in a data processing system consisting of more than two storage subsystems.

[0019] Storage interconnect 50 takes form in a SAN consisting of several devices (e.g., switches, routers, etc.) that operate to transmit data between devices 42-46 including IO transactions to read or write data, it being understood that the term storage interconnect should not be limited thereto. SAN 50 can operate according to any one of a number of different protocols including Fibre Channel, Ethernet, etc. Each of the devices 42-46 includes one or more processors capable of processing data according to instructions of a software component such as a storage manager. As such, each of the devices 42-46 can be considered a computer system.

[0020] Data storage subsystems 44 and 46 may take different forms. For example, data storage subsystems 44 and 46 may take form in object storage devices (OSDs). Unless otherwise noted, each of the data storage subsystems 44 and 46 is implemented as a disk array, it being understood that the term data storage subsystem should not be limited thereto.

Each of the disk arrays 44 and 46 includes several hard disks. Moreover, each of the disk arrays 44 and 46 may include a memory for storing a tag table that will be more fully described below.

[0021] Host 42 includes an application 52 executing on one or more processors. Application 52 may take one of many different forms. For example, application 52 may take form in a database management system (DBMS), customer relationship management software, etc. Application 52 generates IO transactions to read data from or write data to a data volume in response to receiving requests from client computer systems (not shown) coupled to host 42. Host 42 also includes a tag generator 54 executing on one or more processors. Tag generator 54 will be more fully described below.

[0022] Devices 42-46 include a storage manager 60-64, respectively, executing on one or more processors. Each of the storage managers 60-64 is capable of performing many functions. More particularly, each of the storage managers 60-64 is capable of translating IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks of disk arrays 44 and 46. Storage managers 60-64 translate IO transactions according to configuration maps provided thereto. For purposes of illustration, it will be presumed that application 52 is provided with access to a mirrored data volume V having two mirrors, storage objects M0 and M1. It is noted that the present invention can be employed with respect to a data volume consisting of more than two mirrors.

[0023] A visual representation of volume V and mirrors M0 and M1 is found within Figure 4. As shown in Figure 4, each of V, M0, and M1 consist of an array of n_{max} logical memory blocks. It is noted that mirrors M0 and M1 could be formed from underlying storage objects (e.g., logical units or LUNs), or that mirrors M0 and M1 could be created as striped or RAID storage over hard disks or LUNs. It will be presumed, however, that M0 is formed by aggregating physical memory blocks contained within disk d0 (not shown) of disk array 44, and that mirror M1 is formed by aggregating physical memory blocks contained within disk d1 of disk array 46.

[0024] Storage managers 60-64 are provided with storage object descriptions for storage objects V, M0, and M1, respectively. Generally, the storage object descriptions define the relationships between storage objects and their underlying storage objects or hard disks. Moreover, the storage object descriptions may include other information. For example, the storage object descriptions for M0 and M1 indicate that M0 and M1 are mirrors or each other. As such, storage manager 62 knows that a mirror of storage object M0 is contained within storage subsystem 46, and storage manager 64 knows that a mirror of storage object M1 is contained within storage subsystem 44.

[0025] Each of the storage object descriptions may also include a configuration map. A configuration map CMV is provided to storage manager 60 which maps each logical block n of volume V to corresponding blocks n in mirrors M0 and M1. Storage manager 62 is provided with a configuration map CMM0 which maps each logical block n of mirror M0 to a physical memory block x within hard disk d0. Lastly, storage manager 64 is provided with a configuration map CMM1 which maps each logical block n of mirror M1 to a physical memory block y within hard disk d1. Storage manager 60 is capable of translating IO transactions received from application 52 using configuration map CMV. Thus, when application 52 generates a transaction to write a data to block n of volume V, storage manager in response generates first and second transactions for writing a data D to blocks n in mirrors M0 and M1, respectively. Storage manager 62, in response to receiving an IO transaction to write data to block n of mirror M0, uses its configuration map CMM0 to identify the corresponding physical memory block x within disk d0 to which data D is to be written. Likewise, storage manager 64, in response to receiving an IO transaction to write data D to block n of mirror M1, uses its configuration map CMM1 to identify the corresponding physical memory block y within disk d1 to which data D is to be written.

[0026] IO transactions are received by storage manager 60. Tag generator 54, in response to storage manager 60 receiving an IO transaction to write data, generates a tag unique to the IO transaction to write data. This tag is provided to storage manager 60. In one embodiment, the tag may be the count value of a counter. In this embodiment, the count value is incremented by one each time storage manager 60 receives an IO transaction to write data.

[0027] Storage manager 60 accesses its configuration map CMV in response to receiving an IO transaction to write data to logical memory block n of volume V or a range of logical memory blocks beginning with block n. For purposed of explanation, it will be presumed that storage manager 60 receives IO transactions to write data to one logical block n of volume V unless otherwise noted. From configuration map CMV, storage manager 60 learns that logical block n of volume V is mapped to a logical blocks n in mirrors M1 and M0. Accordingly, storage manager 60 generates first and second IO transactions to write data D to block n of mirrors M0 and M1. Each of these first and second IO transactions also includes the unique tag generated by tag generator 54. The first and second IO transactions, including the unique tag, are transmitted by host 42 to disk arrays 44 and 46, respectively, via SAN 50. It is noted that the protocol employed with SAN 50 may have to be modified to accommodate IO transactions that include a unique tag.

[0028] Storage manager 62 is in data communication with nonvolatile memory 70. Likewise, storage manager 64 is in data communication with nonvolatile memory 80. Nonvolatile memories 70 and 80 are configured to store tag tables, such as tag tables 76 and 86, respectively, shown within Figure 5. Tag tables 76 and 86 are accessible by tag table managers 72 and 82, respectively. As will be more fully described below, tag table managers 72 and 82 perform several functions including the creation or deletion entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 may take form in software instructions executing on one or more processors of disk arrays 44 and 46, respectively.

[0029] Tag tables 76 and 86 are associated with mirrors M0 and M1, respectively. It is noted that additional tag tables like tag table 76 or 86 may be contained in memories 70 and 80, where each additional tag table is associated with a mirror of a mirrored data volume provided for access to application 22. However, for purposes of explanation, it will be presumed that memories 70 and 80 store only tag tables 76 and 86, respectively.

[0030] Each tag table 76 and 86 includes a plurality of entries. Each entry stores a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number. Alternatively, each tag table entry stores a tag and a physical block number or a range of physical memory block numbers beginning with a first physical memory block number. For purposes of explanation, it will be presumed that each entry of the tag tables 76 and 86 includes a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number unless otherwise noted. To illustrate, entry 2 of tag table 76 includes a tag equal to 2 and a logical block number equal to 25, and entry 3 of tag table 76 includes a tag equal to 70 and a range of 4 logical blocks beginning with logical memory block number 29.

[0031] Tag tables 76 and 86 track write IO transactions received by storage managers 62 and 64, respectively. Entries in tag tables 76 and 86 are created by tag managers 72 and 82, respectively, each time storage managers 62 and 64, respectively, receive IO transactions to write data to mirrors M0 and M1, respectively. The contents (i.e., the tag and logical block number or range of logical block numbers) of the tag table entries are defined by the IO write transactions received by storage managers 62 and 64. To illustrate, presume storage manager 62 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M0. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 62 provides the tag and the logical block n or range of logical block numbers to receiving the IO transaction, to tag manager 72. Tag table manager 72, in response, creates a new entry in tag table 76. Tag table manager 72 then stores into the newly created entry of table 76 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 62 is written to the physical memory block or blocks in disk d0 allocated to, according to CMM0, the logical block n or range of logical blocks beginning with block n of mirror M0. It is noted that the foregoing process is also employed by tag table manager 82. More particularly, presume storage manager 64 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M1. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 64 provides the tag and the logical block n or range of logical block numbers to receiving the IO transaction, to tag manager 82. Tag table manager 82, in response, creates a new entry in tag table 86. Tag table manager 82 then stores into the newly created entry of table 86 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 64 is written to the physical memory block or blocks in disk d1 allocated to, according to CMM1, the logical block n or range of logical blocks beginning with block n of mirror M1.

[0032] Tag tables 76 and 86 can be used to determine whether mirrors M0 and M1 are in synchronization. If the data processing system 40 operates perfectly, e.g., there is no crash of host 42 or interruption of power to disk arrays 44 and 46, then all IO transactions to write data to mirrors M1 and M0 should reach disk arrays 46 and 44, respectively, and the contents of tag tables 76 and 86 should be identical. However, if due to, for example, crash of host 42 one of two separate but related IO transactions do not reach disk array 44 or 46, then the contents of tag tables 76 and 86 should differ and mirrors M0 and M1 will be out of synchronization. To illustrate, suppose host 42 generates first and second IO transactions for writing data D_{new} to logical block 30 in mirrors M0 and M1. Further, presume that each of these separate IO transactions includes a unique tag 51 generated by generator 54. Lastly, presume that disk array 44 receives the first IO transaction from host 42, but disk array 46 does not receive the second IO transaction. As a result, tag table manager 72 creates a new entry m in tag table 76 which includes tag 51 and block number 30 while no such corresponding tag entry is created in tag table 86. Tag table 76 indicates that logical block 30 of mirror M0 has been updated with data D_{new}. Since table 86 does not include a corresponding entry, it can be said that logical block 30 in mirror M1 has not been updated with data D_{new}, and mirrors M0 and M1 are out of synchronization.

[0033] When mirrors M0 and M1 are out of synchronization, table 76 and 86 can be used to bring mirrors M1 and M0 back into synchronization. Mirrors M0 and M1 will be out of synchronization when an entry is found within table 76 that is not matched in table 86, or when an entry is found in table 86 that is not matched in table 76. When this happens, data is copied from mirror M1 to mirror M0 or from mirror M0 to mirror M1 according to the table entry that lacks the matching entry. To illustrate, tables 76 and 86 are not identical because table 76 includes an entry m consisting of a tag number 51 and logical block 30, and this entry cannot be found within table 86. To bring these mirrors M0 and M1 back into synchronization, data is copied from logical block 30 of mirror M0 to logical block 30 of mirror M1. Thereafter, entry m is deleted from tag table 76. Once this copying process is complete, and assuming there are no other mismatching entries between tables 76 and 86, mirrors M1 and M0 are back in synchronization.

[0034] In addition to creating or deleting entries in tag tables 76 and 86, tag table managers 72 and 82 can operate to (1) identify instances where mirrors M0 and M1 are out of synchronization or (2) delete matching entries in tag tables 76 and 86. Figure 6 illustrates these operational aspects of tag table manager 72. The operational aspects shown in Figure 6 can be initiated at any time. For example, the process shown in Figure 6 can be initiated after failure of host 42.

[0035] After the process is started in step 90, tag table manager 72 selects an entry in tag table 76 as shown in step 94 and then sends a message to tag table manager 82 asking if there is a corresponding match within tag table 86. The message sent to tag table manager 82 identifies the tag and logical block number or range of logical block numbers of the tag selected in step 92. Tag table manager 82 accesses tag table 86 using tag and logical block number or range of logical block numbers of the entry selected in step 92 that is provided in the message sent by tag table manager 72, and determines if a match exists as shown in step 94. If tag table manager 82 finds an entry in table 86 that matches the entry selected in table 76 (i.e., an entry in table 86 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in step 92), then tag table manager deletes the matching entry in table 86 and sends a reply message to tag table manager 72 indicating that table 86 did include a match to the entry selected in step 92. Tag manager 72, in response to receiving the message from tag table manager 82 that a matching entry did exist, deletes the entry selected in step 92. The entry deletion by tag managers 72 and 82 is shown in step 96. However, if tag table 86 does not have a match to the entry selected in table 76, then a message to that effect is sent back to tag table manager 72 and the process proceeds to step 100. In step 100, data is copied from mirror M0 to mirror M1. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M0 that are identified by the entry selected in step 92, is copied to mirror M1 at the logical memory block or range of logical memory blocks identified by the entry selected in step 92. For example, assuming the selected entry in step 92 is entry m of table 76, the data contents of logical block 30 of mirror M0 is copied to logical block 30 of mirror M1 in step 100. After copying data from mirror M0 to mirror M1in step 100, tag table manager 72 deletes the entry selected in table 76 as shown in step 104. Thereafter, as shown in step 104, tag table manager 72 checks tag table 76 to see if any tag entries remain. If an entry remains, then it is selected in step 92, and steps 94-102 are repeated. If no additional entries remain in table 76 at step 104, the process ends.

[0036] It is noted that tag table manager 82 operates under a process similar to that shown in Figure 6. In particular, tag table manager 82 selects an entry in tag table 86 and sends a message to tag table manager 72 asking if there is a corresponding match within tag table 76. The message sent to tag table manager 72 identifies the tag and logical block number or range of logical block numbers of the tag selected in table 86. Tag table manager 72 accesses tag table 76 using tag and logical block number or range of logical block numbers of the selected entry of table 86 provided in the message sent by tag table manager 82, and determines if a match exists in table 76. If tag table manager 72 finds an entry in table 76 that matches the entry selected in table 86 (i.e., an entry in table 76 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in table 86), then tag table manager 72 deletes the matching entry in table 76 and sends a reply message to tag table manager 82 indicating that table 76 did include a match to the entry selected in table 86. Tag manager 82, in response to receiving the message from tag table manager 72 that a matching entry did exist in table 76, deletes the selected entry in table 86. However, if tag table 76 does not have a match to the entry selected in table 86, then a message to that effect is sent to tag table manager 82 and data is copied from mirror M1 to mirror M0. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M1 that are identified by the entry selected in table 86 is copied to the logical memory block or range of logical memory blocks identified by the entry selected in table 86 but contained in mirror M0. After copying data from mirror M1 to mirror M0, tag table manager 82 deletes the selected entry in table 86. Thereafter, tag table manager 82 checks tag table 86 to see if any tag entries remain. If an entry remains, then it is selected, and the foregoing steps are repeated. If no additional entries remain in table 86, the process ends.

[0037] As noted above, storage subsystems may take form in OSDs. In this embodiment, tag tables, such as tag tables 76 and 86, my have to be modified to include an object number and information indicating whether the corresponding write operation is overwriting existing data of the storage object or whether the corresponding write operation extends the file length.

[0038] Although the present invention has been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein. On the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the scope of the invention as defined by the appended claims.

WE CLAIM:

1. A method comprising:

a computer system generating first and second write transactions;

wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;

the computer system transmitting the fist and second write transactions to first and second storage devices, respectively.

2. The method of claim 1 further comprising:

the computer system generating third and fourth write transactions;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;

the computer system transmitting the third and fourth write transactions to the first and second storage devices, respectively.

3. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a logical block of a first storage object;

the second write transaction comprises data D to be written to a logical block of a second storage object.

4. The method of claim 3 further comprising:

the first storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory;

the second storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

5. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

6. The method of claim 5 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an identity of the range of logical blocks of the first storage object where data D is to be written, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an identity of the range of logical blocks in the second storage object where data D is to be written, wherein the second tag table is stored in second memory.

7. The method of claim 4 further comprising comparing the contents of one entry in the first tag table with the contents of entries in the second tag table.

8. The method of claim 7 further comprising copying data from physical memory allocated to a logical block number identified in the one entry to physical memory allocated to the second storage object if the second table lacks an entry with contents matching the contents of the one entry.

9. The method of claim 7 further comprising deleting the one entry in the first table if the second table contains an entry with contents that match the contents of the one entry.

10. The method of claim 9 further comprising deleting the entry in the second table with contents that match the contents of the one entry.

11. The method of claim 1 wherein the first tag is identical to the second tag.

12. The method of claim 1 further comprising:

the computer system generating a write transaction to write data to a logical block of a data volume;

the computer system incrementing a counter in response to generating the write transaction;

the computer system generating the first and second tags as a function of an output of the incremented counter.

13. The method of claim 1 wherein the first and second storage devices comprise first and second object storage devices.

14. The method of claim 11 wherein:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

17. The method of claim 16 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an indication that data D is to be stored in the extension of the first storage object, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an indication that data D is to be stored in the extension of the second storage object, wherein the second tag table is stored in second memory.

18. A method comprising:

a computer system generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

the computer system transmitting the transaction to a storage device.

19. The method of claim 18 wherein the storage device stores a replication of the storage object.

ATTORNEY/CLIENT PRIVILEGED AND CONFIDENTIAL

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIV E VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha Valiveti
Dhanesh Joshi

ABSTRACT OF THE DISCLOSURE

[0039]

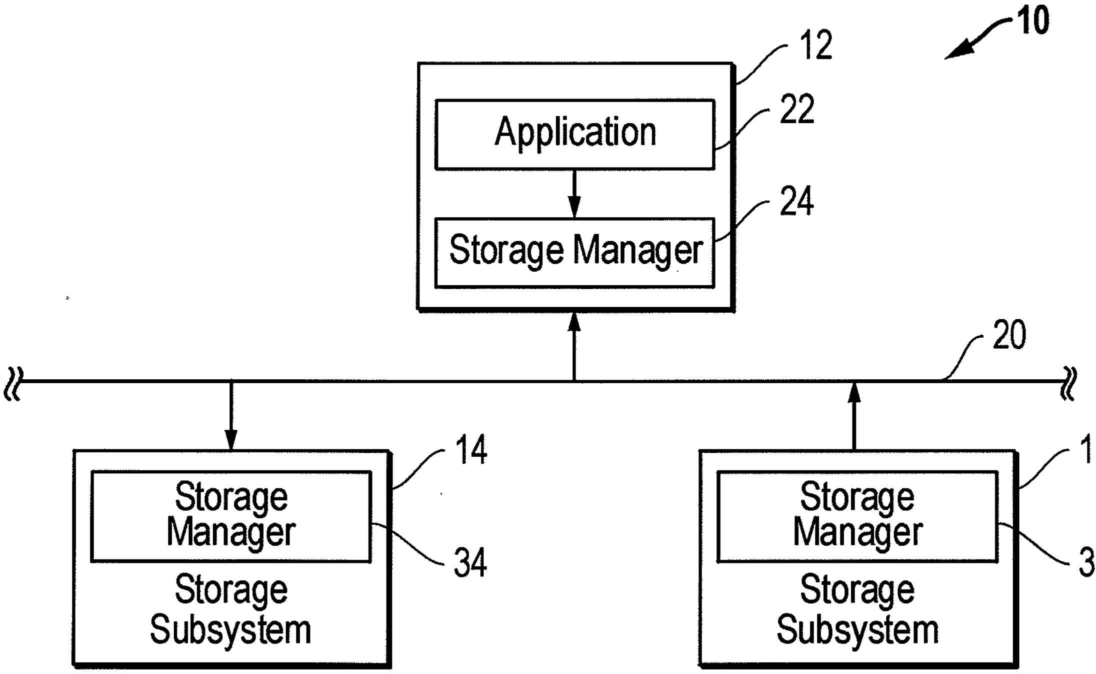

10
12
Application
22
24
Storage Manager
20
14
Storage Manager
Storage Subsystem
34
16
Storage Manager
Storage Subsystem
36

FIG. 1

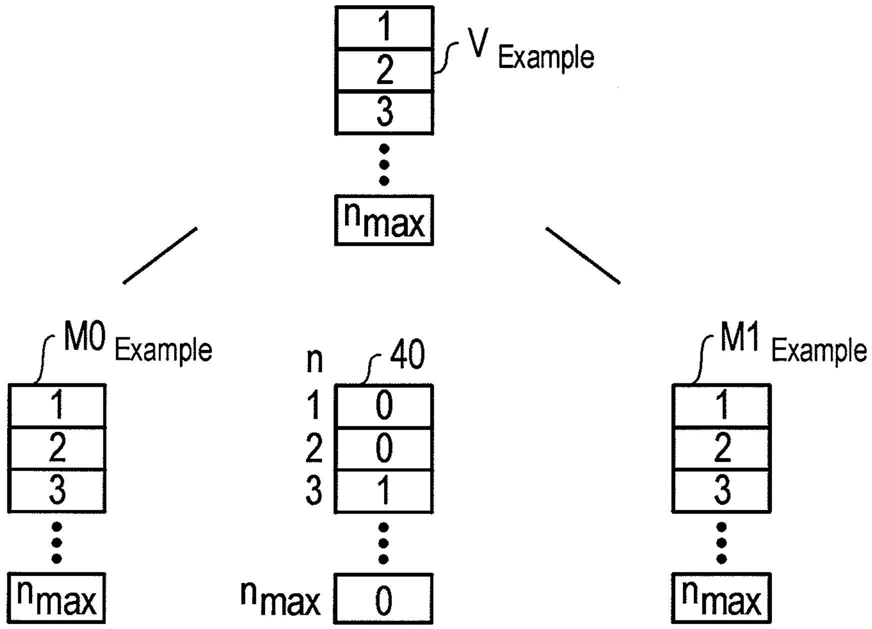

1
2
3
n max
V Example
M0 Example
1
2
3
n max
n
40
1 0
2 0
3 1
n max 0
M1 Example
1
2
3
n max

FIG. 2

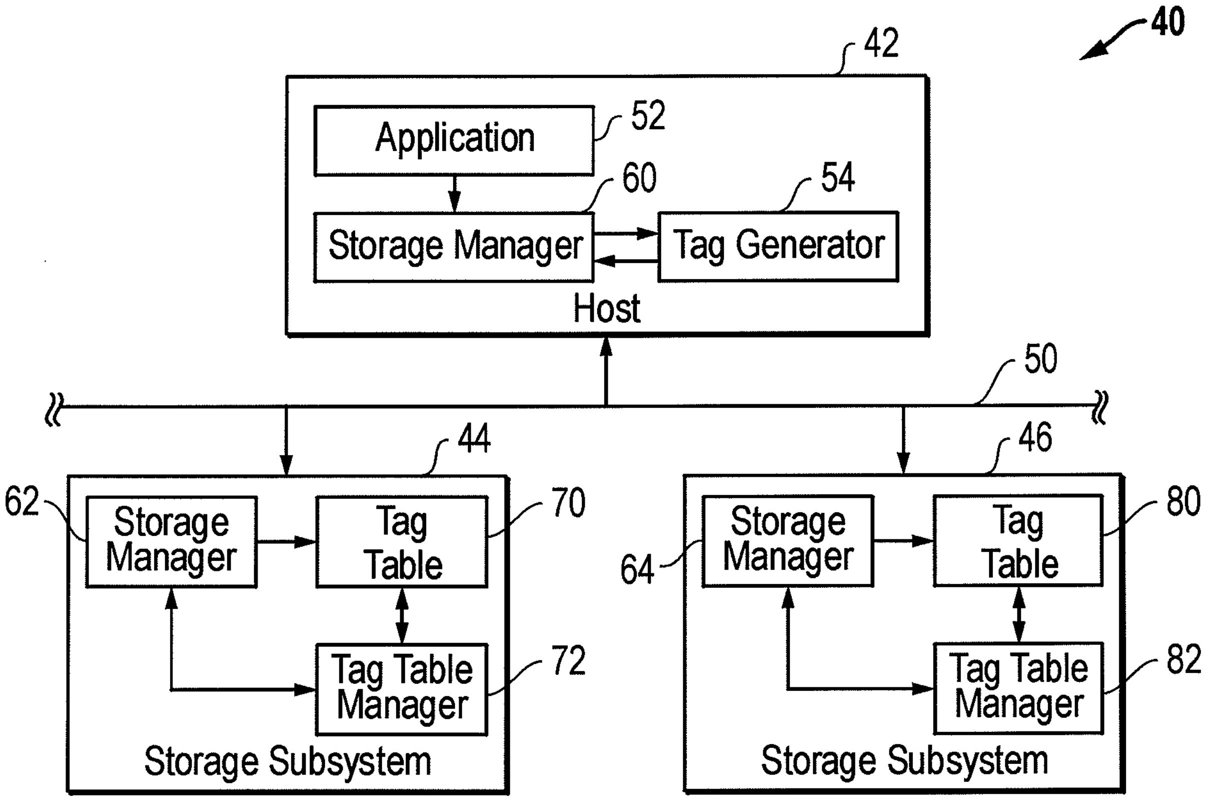

40
42
Application
52
60
54
Storage Manager
Tag Generator
Host
50
44
46
62
Storage Manager
Tag Table
70
Tag Table Manager
72
Storage Subsystem
64
Storage Manager
Tag Table
80
Tag Table Manager
82
Storage Subsystem

FIG. 3

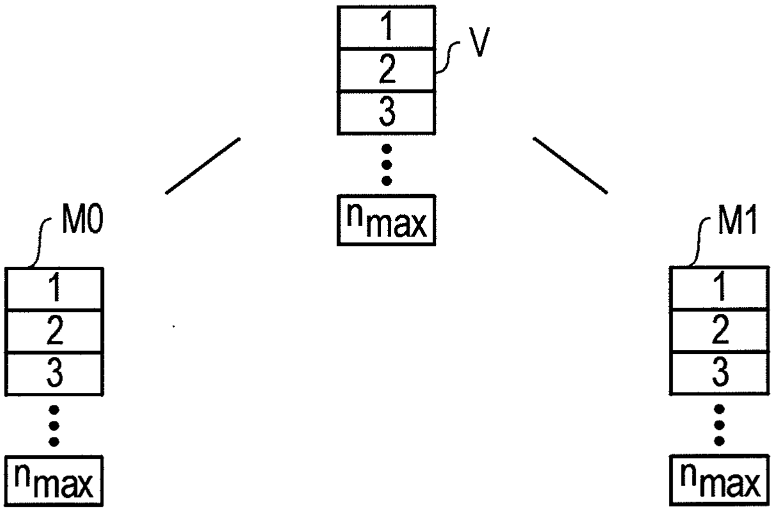
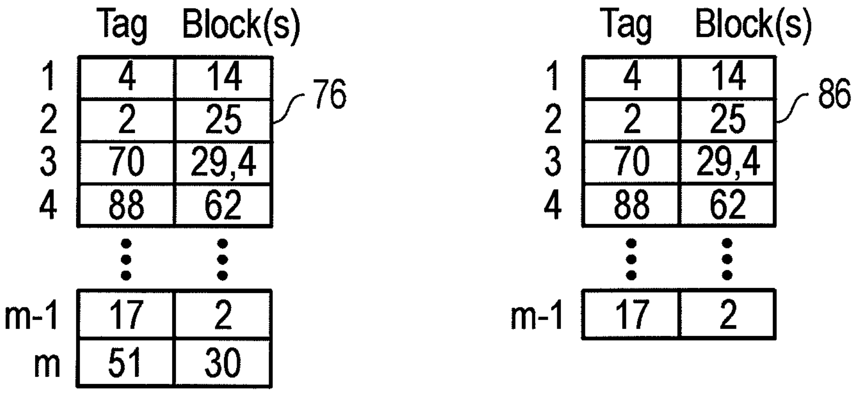

1
2
3
n_{max}
V
M0
1
2
3
n_{max}
M1
1
2
3
n_{max}

FIG. 4



76

	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
	⋮	⋮
m-1	17	2
m	51	30

86

	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
	⋮	⋮
m-1	17	2

FIG. 5

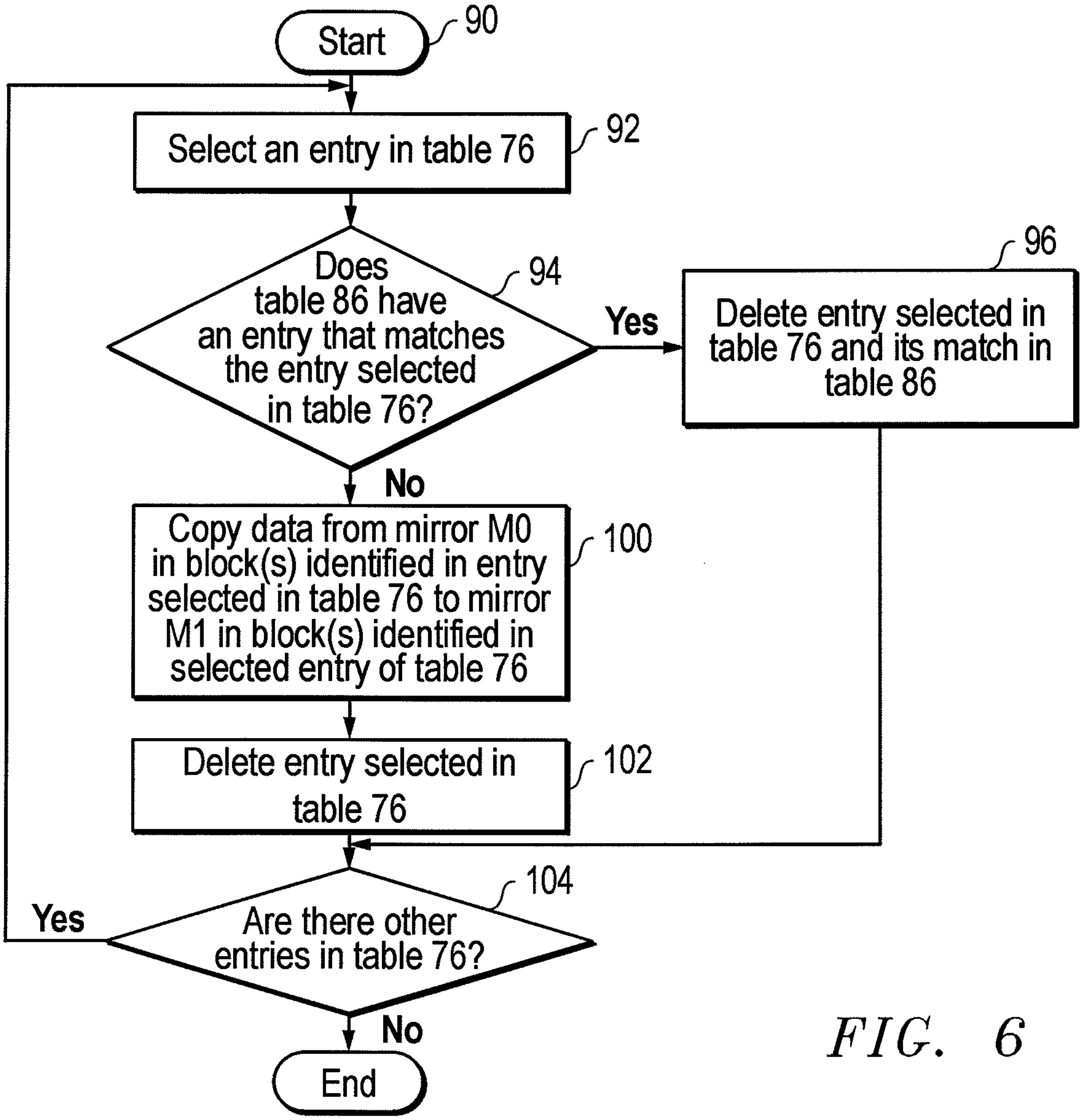

Start
90
Select an entry in table 76
92
Does table 86 have an entry that matches the entry selected in table 76?
94
Yes
Delete entry selected in table 76 and its match in table 86
96
No
Copy data from mirror M0 in block(s) identified in entry selected in table 76 to mirror M1 in block(s) identified in selected entry of table 76
100
Delete entry selected in table 76
102
Are there other entries in table 76?
104
Yes
No
End

FIG. 6

EXHIBIT B

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:54 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) First Draft Patent Application
Attachments: VRT0120US-Patent Application.DOC; VRT0120US.pdf

From: Eric Stephenson
Sent: Friday, February 20, 2004 9:53 AM
To: 'Ron Karr'; 'Narasimha Valiveti'; Ramana Jonnala (ramana@veritas.com); Dhanesh Joshi (dhanesh@veritas.com)
Cc: julie.stephenson@veritas.com; Roz Donaldson; Anne Castle; Sam Campbell; Eric Stephenson
Subject: VRTS 0655 (VRT0120US) First Draft Patent Application

Greetings,

Attached please find a first draft of the above identified patent application.

Please review the patent application including the claims. After your review, I will make any changes to the application you deem necessary.

Please note that I have left the summary and abstract sections blank until the claims are set.

Thanks,

Eric

Eric Stephenson
Campbell Stephenson Ascolese LLP
4807 Spicewood Springs Road
Suite 4-201
Austin, Texas 78759
(512) 439-5093 Direct
(512) 439-5099 Fax
estephenson@csapatent.com

THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR THE PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED ABOVE. This message and the information contained herein is a confidential, attorney-client privileged communication. If you are not the intended recipient, you have received this document in error, and any review, dissemination, distribution, or copying of this message is strictly prohibited. If you are an unintended recipient, please notify the sender immediately and delete this message and any copies thereof. Thank you.

EXHIBIT C

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:54 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) First Draft Patent Application

```
-----Original Message-----
From: Ronald S. Karr [mailto:tron@veritas.com]
~~Sent: Friday, February 20, 2004 8:17 PM~~
To: Eric Stephenson
Cc: Ron Karr; Narasimha Valiveti; ramana@veritas.com; dhanesh@veritas.com
Subject: Re: VRTS 0655 (VRT0120US) First Draft Patent Application

I reviewed all the claims.

You use the word "fist" instead of "first" in claim 1.

In claims 8 and later you mention the word "match" to suggest a recognized match of the
tags, but that isn't exactly spelled out.  Claim 7 seems to be the algorithm for finding
the match, but you don't use the word "match".

Is claim 12 your attempt to add mirroring into the claims?
If so, don't you need to say that the first and second write tags relate to first and
second write transctions generated as a result of the logical volume write transaction?

Other than that, I am so far happy with the claims.  I have no idea when I will have time
to thoroughly read through the rest of the patent.
--
        Ronald S. Karr
        tron |-<=>-|    tron@veritas.com




>    -----Original Message-----
>    From: Eric Stephenson
>    Sent: Friday, February 20, 2004 9:53 AM
>    To: 'Ron Karr'; 'Narasimha Valiveti'; Ramana Jonnala
>(ramana@veritas.com); Dhanesh Joshi (dhanesh@veritas.com)
>    Cc: julie.stephenson@veritas.com; Roz Donaldson; Anne Castle;
>Sam Campbell; Eric Stephenson
>    Subject: VRTS 0655 (VRT0120US) First Draft Patent Application
>
>
>    Greetings,
>
>
>    Attached please find a first draft of the above identified
>patent application.
>
>    Please review the patent application including the claims. After
```

>your review, I will make any changes to the application you deem >necessary.
>
> Please note that I have left the summary and abstract sections
>blank until the claims are set.
>
> Thanks,
>
> Eric
>
> Eric Stephenson
> Campbell Stephenson Ascolese LLP
> 4807 Spicewood Springs Road
> Suite 4-201
> Austin, Texas 78759
> (512) 439-5093 Direct
> (512) 439-5099 Fax
> estephenson@csapatent.com
>
>
>
> THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR
>THE PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED >ABOVE. This message and the information contained herein is a >confidential, attorney-client privileged communication. If you are not >the intended recipient, you have received this document in error, and >any review, dissemination, distribution, or copying of this message is >strictly prohibited. If you are an unintended recipient, please notify >the sender immediately and delete this message and any copies thereof.
>Thank you.

EXHIBIT D

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:55 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Second Draft Patent Application

---

From: Eric Stephenson
Sent: Monday, February 23, 2004 5:37 AM
To: 'Ron Karr'; 'Narasimha Valiveti'; 'Ramana Jonnala (ramana@veritas.com)'; 'Dhanesh Joshi (dhanesh@veritas.com)'
Cc: 'julie.stephenson@veritas.com'; Roz Donaldson; Anne Castle; Sam Campbell
Subject: VRTS 0655 (VRT0120US) Second Draft Patent Application

Greetings,

Attached please find a second draft of the above identified patent application. This draft includes changes to the claims requested by Ron Karr.

Please review the patent application including the claims. After your review, I will make any changes to the application you deem necessary.

Please note that we seek to file this application by Thursday, February 26.
Thanks,

Eric

Eric Stephenson
Campbell Stephenson Ascolese LLP
4807 Spicewood Springs Road
Suite 4-201
Austin, Texas 78759
(512) 439-5093 Direct
(512) 439-5099 Fax
estephenson@csapatent.com

THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR THE PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED ABOVE. This message and the information contained herein is a confidential, attorney-client privileged communication. If you are not the intended recipient, you have received this document in error, and any review, dissemination, distribution, or copying of this message is strictly prohibited. If you are an unintended recipient, please notify the sender immediately and delete this message and any copies thereof. Thank you.

EXHIBIT E

ATTORNEY/CLIENT PRIVILEGED AND CONFIDENTIAL

THIS DRAFT PATENT APPLICATION CONTAINS INFORMATION CONFIDENTIAL TO *VERITAS Software Corporation* AND INCLUDES PRIVILEGED, ATTORNEY-CLIENT COMMUNICATIONS

EXPRESS MAIL LABEL NO.:

(EV 304738055 US)

DRAFT

8/10/2009; 3:07:12 PM

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

BACKGROUND OF THE INVENTION

[0001] Large scale data processing systems typically include several data storage subsystems (e.g., disk arrays) each containing many physical storage devices (e.g., hard disks) for storing critical data. Data processing systems often employ storage management systems to aggregate these physical storage devices to create highly reliable data storage. There are many types of highly reliable storage. Mirrored volume storage is an example. Mirrored volume storage replicates data over two or more mirrors of equal size. A logical memory block n of a mirrored volume maps to the same logical memory block n of each mirror. In turn, each logical memory block n of the mirrors map directly or indirectly to one or more physical memory blocks of one or more physical devices. Mirrored volumes provide data redundancy. If an application is unable to access data of one, a duplicate of the data sought should be available in an alternate mirror.

[0002] Figure 1 illustrates relevant components of an exemplary data processing system 10 that employs a two-way mirrored volume. While the present invention will be described with reference to a two-way mirrored volume, the present invention should not be limited thereto. The present invention may find use with other types of redundant storage including, for example, a three-way mirrored storage volume. Data processing system 10 includes a host (e.g., server computer system) 12 coupled to data storage subsystems 14 and 16 via storage interconnect 20. For purposes of explanation, storage interconnect 20 will take form in a storage area network (SAN) it being understood that the term storage interconnect should not be limited thereto. SAN 20 may include devices (e.g., switches, routers, hubs, etc.) that cooperate to transmit input/output (IO) transactions between host 12 and storage subsystems 14 and 16.

[0003] Each of the data storage subsystems 14 and 16 includes several physical storage devices. For purposes of explanation, data storage subsystems 14 and 16 are assumed to include several hard disks. The term physical storage device should not be limited to hard disks. Data storage subsystems 14 and 16 may take different forms. For example, data storage subsystem 14 may consist of "Just a Bunch of Disks" (JBOD) connected to an array controller card. Data storage subsystem 16 may consist of a block server appliance. For purposes of explanation, each of the data storage subsystems 14 and 16 will take form in an intelligent disk array, it being understood that the term data storage subsystem should not be limited thereto.

[0004] As noted, each of the disk arrays 14 and 16 includes several hard disks. The hard disk is the most popular permanent storage device currently used. A hard disk's total storage capacity is divided into many small chunks called physical memory blocks. For example, a 10 GB hard disk contains 20 million physical memory blocks, with each block able to hold 512 bytes of data. Any random physical memory block can be written to or read from in about the same time, without first having to read or write other physical memory blocks. Once written, a physical memory block continues to hold data even after the hard disk is powered down. While hard disks in general are reliable, they are subject to occasional failure. Data systems employ data redundancy schemes such as mirrored volumes to protect against occasional failure of a hard disk.

[0005] Host 12 includes an application 22 executing on one or more processors. Application 22 generates IO transactions to access critical data in response to receiving requests from client computer systems (not shown) coupled to host 12. In addition to application 22, host 12 includes a storage manager 24 executing on one or more processors. Volume Manager^{TM} provided by VERITAS Software Corporation of Mountain View, California, is an exemplary storage manager. Although many of the examples described herein will emphasize virtualization architecture and terminology associated with the VERITAS Volume Manager™, the software and techniques described herein can be used with a variety of different storage managers and architectures.

[0006] Storage managers can perform several functions. More particularly, storage managers can create storage objects (also known as virtualized disks) by aggregating hard disks such as those of disk arrays 14 and 16, underlying storage objects, or both. A storage object is an abstraction. Figure 2 shows a visual representation of exemplary storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ created for use in data processing system 10. Each of the storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ in Figure 2 consists of an array of n_{max} logical memory blocks that store or are configured to store data. While it is said that a logical memory block stores or is configured to store data, in reality the data is stored in one or more physical memory blocks of hard disks allocated directly or indirectly to the logical memory block.

[0007] Storage objects aggregated from hard disks can themselves be aggregated to form storage objects called logical data volumes. Logical data volumes are typically presented for direct or indirect use by an application such as application 22 executing on host 12. Thus, application 22 generates IO transactions to read data from or write data to one or more logical memory blocks of a data volume not knowing that the data volume is an aggregation of underlying storage objects, which in turn are aggregations of hard disks. Properties of storage objects depend on how the underlying storage objects or hard disks are aggregated.

In other words, the method of aggregation determines the storage object type. In theory, there are a large number of possible methods of aggregation. The more common forms of aggregation include concatenated storage, striped storage, RAID storage, or mirrored storage. A more thorough discussion of how storage objects or hard disks can be aggregated can be found within Dilip M. Ranade [2002], "Shared Data Clusters" Wiley Publishing, Inc., which is incorporated herein by reference in its entirety.

[0008] Mirrored volumes provide highly reliable access to critical data. $V_{Example}$ of Figure 2 is an exemplary two-way mirrored volume. $V_{Example}$ was created by aggregating underlying storage objects (hereinafter mirrors) $M0_{Example}$ and $M1_{Example}$. Mirrors $M0_{Example}$ and $M1_{Example}$ were created by concatenating physical memory blocks from hard disks $d0_{Example}$ and $d1_{Example}$ (not shown) in disk arrays 14 and 16, respectively.

[0009] Storage managers can create storage object descriptions that describe the relationship between storage objects and their underlying storage objects or hard disks. These storage object descriptions typically include configuration maps. It is noted that storage object descriptions may include other information such as information indicating that a storage object is a snapshot copy of another storage object.

[0010] A configuration map maps a logical memory block of a corresponding storage object to one or more logical memory blocks of one or more underlying storage objects or to one or more physical memory blocks of one or more hard disks. To illustrate, configuration maps $CMV_{Example}$, $CMM0_{Example}$, and $CMM1_{Example}$ are created for mirrored volume $V_{Example}$ and underlying mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Configuration map $CMV_{Example}$ maps each logical memory block n of $V_{Example}$ to logical memory blocks n of mirrors $M0_{Example}$ and $M1_{Example}$. Configuration map $CMM0_{Example}$ maps each logical memory block n of mirror $M0_{Example}$ to a physical memory block x in hard disk $d0_{Example}$, while configuration map $CMM1_{Example}$ maps each logical memory block n of mirror $M1_{Example}$ to a physical memory block y in hard disk $d1_{Example}$. Configuration map $CMV_{Example}$ can be provided for use by storage manager 24, while configuration maps $CMM0_{Example}$ and $CMM1_{Example}$ can be provided for use by storage managers 34 and 36 executing on one or more processors in disk arrays 14 and 16, respectively.

[0011] Storage managers use configuration maps to translate IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks. To illustrate, presume an IO transaction is generated by application 22 to write data D to logical memory block 3 of data volume $V_{Example}$. This IO transaction is received directly or indirectly by storage manager 24. In turn, storage manager 24 accesses configuration map $CMV_{Example}$ and learns that logical memory block 3 is mapped to logical block 3 in both mirrors $M0_{Example}$ and $M1_{Example}$. It is noted storage manager may not receive the exact IO transaction generated by application 22. However, the transaction storage manager 24 receives will indicate that data D is to be written to block 3 of mirrored volume $V_{Example}$.

[0012] Storage manager 24 then generates first and second IO transactions to write data D to logical blocks 3 and mirrors $M0_{Example}$ and $M1_{Example}$, respectively. The IO transactions generated by storage manager 24 are transmitted to disk arrays 14 and 16, respectively, via SAN 20. Storage managers 34 and 36 of disk arrays 14 and 16, respectively receive directly or indirectly, the IO transactions sent by storage manager 24. It is noted that the IO transactions received by storage managers 34 and 36 may not be the exact IO transactions generated and sent by storage manager 24. Nonetheless, storage managers 34 and 36 will each receive an IO transaction to write data D to logical block 3 in mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Storage manager 34, in response to receiving the IO transaction, accesses configuration map $CMM0_{Example}$ to learn that block 3 of mirror $M0_{Example}$ is mapped to, for example, physical memory block 200 within hard disk d0. In response, a transaction is generated to write data D to physical memory block 200 within disk d0. Storage manager 36 accesses configuration map $CMM1_{Example}$ to learn that logical block 3 of mirror $M1_{Example}$ is mapped to, for example, physical memory block 300 within hard disk d1. In response, an IO transaction is generated for writing data D to physical memory block 300 within hard disk d1.

[0013] As noted above, while hard disks are reliable, hard disks are subject to failure. Data may be inaccessible within a failed hard disk. For this reason and others, administrators create mirrored data volumes. Unfortunately, data within mirrors of a mirrored volume may get into an inconsistent state if, for example, there is a server crash, storage power failure, or other problem which prevents data from being properly written to a hard disk. Consider the exemplary mirrored volume $V_{Example}$. Presume storage manager 24 generates first and second IO transactions to write data D to logical blocks 3 in mirrors $M0_{Example}$ and $M1_{Example}$ in response to the IO transaction generated by application 22. Further, presume host 12 may fail after the first IO transaction is transmitted to disk array 14, but before the second IO transaction is transmitted to disk array 16. As a result, data D is written to physical memory block 200 within disk d0 of disk array 14, but data D is not written to physical memory block 300 within hard disk d1. When host 12 is restarted and exemplary volume $V_{Example}$ is made available again to application 22, mirrors $MO_{Example}$ and $M1_{Example}$ are said to be out of sync. In other words, mirrors $M0_{Example}$ and $M1_{Example}$ are no longer identical since at least block 3 in each contain different data. An IO transaction to read from logical memory block 3 of mirrored volume $V_{Example}$ could return either old or new data depending on whether the data is read from physical memory block 200 of hard disk d0 or physical memory block 300 of hard disk d1. Mirrors $MO_{Example}$ and $M1_{Example}$ should be resychronized before either is accessed again.

[0014] A brute force method to resynchronize mirrors $M0_{Example}$ and $M1_{Example}$ is simply to presume that one mirror (e.g., $M0_{Example}$) contains correct data and copy the contents of the one mirror to the other (e.g., $M1_{Example}$). It can take hours to resynchronize using this method. A smarter resynchronization technique is possible, but it requires some preparation. This alternate technique involves using what is called a dirty region map. Figure 2 illustrates a visual representation of an exemplary dirty region map 40 consisting of n_{max} entries corresponding to the n_{max} logical memory blocks within mirrors $M0_{Example}$ and $M1_{Example}$. Each entry of the dirty region map 40 indicates whether the corresponding blocks in mirrors are considered synchronized. For example, if entry n is set to logical 1, then blocks n in the mirrors are considered out of synchronization, and if n is set to logical 0, blocks n in the mirrors are considered in synchronization. Entry n in dirty region map 40 is set to logical 1 when the application 22 generates a transaction for writing data to logical block n of volume

$V_{Example}$. Entry n is maintained in the logical 1 state until acknowledgement is received from both disk arrays 14 and 16 that data D has been written successfully to the physical memory blocks allocated to logical memory block n. However, if, using the example above, the first IO transaction to write data D to block 3 of mirror $M0_{Example}$ succeeds while the second IO transaction to write to block 3 of $M1_{Example}$ fails, then entry 3 and dirty region map 40 will be maintained as a logical 1 indicating that logical blocks 3 in the mirrors are out of synchronization. Mirrors $M0_{Example}$ and $M1_{Example}$ can be resynchronized by copying data from $M0_{Example}$ to mirror $M1_{Example}$, but for only logical memory blocks corresponding to dirty region map entries set to logical 1.

SUMMARY OF THE INVENTION

[0015] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

Fig. 1 illustrates a block diagram of a data system in which a logical data volume may be created;

Fig. 2 illustrates a visual representation of the storage objects created in the data system of Fig. 1;

Fig. 3, illustrates a block diagram of a data system capable of implementing one embodiment of the present invention;

Figs. 4 shows visual representations of a mirrored data volume employed in the data system of Fig. 3;

Fig. 5 is a visual representation of exemplary tag tables employed in the disk arrays of Fig. 3;

Fig.6 is a flow chart illustrating relevant aspects of managing the tag tables shown in Fig. 5.

[0017] The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION

[0018] Figure 3 illustrates in block diagram form, relevant components of an exemplary data processing system 40 capable of employing one embodiment of the present invention. Data processing system 40 includes a host (e.g., server computer system) 42 coupled to data storage subsystems 44 an 46 via storage interconnect 50. The present invention should not be limited to use in a data processing system consisting of only two storage subsystems. The present invention may be employed in a data processing system consisting of more than two storage subsystems.

[0019] Storage interconnect 50 takes form in a SAN consisting of several devices (e.g., switches, routers, etc.) that operate to transmit data between devices 42-46 including IO transactions to read or write data, it being understood that the term storage interconnect should not be limited thereto. SAN 50 can operate according to any one of a number of different protocols including Fibre Channel, Ethernet, etc. Each of the devices 42-46 includes one or more processors capable of processing data according to instructions of a software component such as a storage manager. As such, each of the devices 42-46 can be considered a computer system.

[0020] Data storage subsystems 44 and 46 may take different forms. For example, data storage subsystems 44 and 46 may take form in object storage devices (OSDs). Unless otherwise noted, each of the data storage subsystems 44 and 46 is implemented as a disk array, it being understood that the term data storage subsystem should not be limited thereto. Each of the disk arrays 44 and 46 includes several hard disks. Moreover, each of the disk arrays 44 and 46 may include a memory for storing a tag table that will be more fully described below.

[0021] Host 42 includes an application 52 executing on one or more processors. Application 52 may take one of many different forms. For example, application 52 may take form in a database management system (DBMS), customer relationship management software, etc. Application 52 generates IO transactions to read data from or write data to a data volume in response to receiving requests from client computer systems (not shown) coupled to host 42. Host 42 also includes a tag generator 54 executing on one or more processors. Tag generator 54 will be more fully described below.

[0022] Devices 42-46 include a storage manager 60-64, respectively, executing on one or more processors. Each of the storage managers 60-64 is capable of performing many functions. More particularly, each of the storage managers 60-64 is capable of translating IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks of disk arrays 44 and 46. Storage managers 60-64 translate IO transactions according to configuration maps provided thereto. For purposes of illustration, it will be presumed that application 52 is provided with access to a mirrored data volume V having two mirrors, storage objects M0 and M1. It is noted that the present invention can be employed with respect to a data volume consisting of more than two mirrors.

[0023] A visual representation of volume V and mirrors M0 and M1 is found within Figure 4. As shown in Figure 4, each of V, M0, and M1 consist of an array of n_{max} logical memory blocks. It is noted that mirrors M0 and M1 could be formed from underlying storage objects (e.g., logical units or LUNs), or that mirrors M0 and M1 could be created as striped or RAID storage over hard disks or LUNs. It will be presumed, however, that M0 is formed by aggregating physical memory blocks contained within disk d0 (not shown) of disk array 44, and that mirror M1 is formed by aggregating physical memory blocks contained within disk d1 of disk array 46.

[0024] Storage managers 60-64 are provided with storage object descriptions for storage objects V, M0, and M1, respectively. Generally, the storage object descriptions define the relationships between storage objects and their underlying storage objects or hard disks. Moreover, the storage object descriptions may include other information. For example, the storage object descriptions for M0 and M1 indicate that M0 and M1 are mirrors or each other. As such, storage manager 62 knows that a mirror of storage object M0 is contained within storage subsystem 46, and storage manager 64 knows that a mirror of storage object M1 is contained within storage subsystem 44.

[0025] Each of the storage object descriptions may also include a configuration map. A configuration map CMV is provided to storage manager 60 which maps each logical block n of volume V to corresponding blocks n in mirrors M0 and M1. Storage manager 62 is provided with a configuration map CMM0 which maps each logical block n of mirror M0 to a physical memory block x within hard disk d0. Lastly, storage manager 64 is provided with a configuration map CMM1 which maps each logical block n of mirror M1 to a physical memory block y within hard disk d1. Storage manager 60 is capable of translating IO transactions received from application 52 using configuration map CMV. Thus, when application 52 generates a transaction to write a data to block n of volume V, storage manager in response generates first and second transactions for writing a data D to blocks n in mirrors M0 and M1, respectively. Storage manager 62, in response to receiving an IO transaction to write data to block n of mirror M0, uses its configuration map CMM0 to identify the corresponding physical memory block x within disk d0 to which data D is to be written. Likewise, storage manager 64, in response to receiving an IO transaction to write data D to block n of mirror M1, uses its configuration map CMM1 to identify the corresponding physical memory block y within disk d1 to which data D is to be written.

[0026] IO transactions are received by storage manager 60. Tag generator 54, in response to storage manager 60 receiving an IO transaction to write data, generates a tag unique to the IO transaction to write data. This tag is provided to storage manager 60. In one embodiment, the tag may be the count value of a counter. In this embodiment, the count value is incremented by one each time storage manager 60 receives an IO transaction to write data.

[0027] Storage manager 60 accesses its configuration map CMV in response to receiving an IO transaction to write data to logical memory block n of volume V or a range of logical memory blocks beginning with block n. For purposed of explanation, it will be presumed that storage manager 60 receives IO transactions to write data to one logical block n of volume V unless otherwise noted. From configuration map CMV, storage manager 60 learns that logical block n of volume V is mapped to a logical blocks n in mirrors M1 and M0. Accordingly, storage manager 60 generates first and second IO transactions to write data D to block n of mirrors M0 and M1. Each of these first and second IO transactions also includes the unique tag generated by tag generator 54. The first and second IO transactions, including the unique tag, are transmitted by host 42 to disk arrays 44 and 46, respectively, via SAN 50. It is noted that the protocol employed with SAN 50 may have to be modified to accommodate IO transactions that include a unique tag.

[0028] Storage manager 62 is in data communication with nonvolatile memory 70. Likewise, storage manager 64 is in data communication with nonvolatile memory 80. Nonvolatile memories 70 and 80 are configured to store tag tables, such as tag tables 76 and 86, respectively, shown within Figure 5. Tag tables 76 and 86 are accessible by tag table managers 72 and 82, respectively. As will be more fully described below, tag table managers 72 and 82 perform several functions including the creation or deletion entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 may take form in software instructions executing on one or more processors of disk arrays 44 and 46, respectively.

[0029] Tag tables 76 and 86 are associated with mirrors M0 and M1, respectively. It is noted that additional tag tables like tag table 76 or 86 may be contained in memories 70 and 80, where each additional tag table is associated with a mirror of a mirrored data volume provided for access to application 22. However, for purposes of explanation, it will be presumed that memories 70 and 80 store only tag tables 76 and 86, respectively.

[0030] Each tag table 76 and 86 includes a plurality of entries. Each entry stores a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number. Alternatively, each tag table entry stores a tag and a physical block number or a range of physical memory block numbers beginning with a first physical memory block number. For purposes of explanation, it will be presumed that each entry of the tag tables 76 and 86 includes a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number unless otherwise noted. To illustrate, entry 2 of tag table 76 includes a tag equal to 2 and a logical block number equal to 25, and entry 3 of tag table 76 includes a tag equal to 70 and a range of 4 logical blocks beginning with logical memory block number 29.

[0031] Tag tables 76 and 86 track write IO transactions received by storage managers 62 and 64, respectively. Entries in tag tables 76 and 86 are created by tag managers 72 and 82, respectively, each time storage managers 62 and 64, respectively, receive IO transactions to write data to mirrors M0 and M1, respectively. The contents (i.e., the tag and logical block number or range of logical block numbers) of the tag table entries are defined by the IO write transactions received by storage managers 62 and 64. To illustrate, presume storage manager 62 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M0. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 62 provides the tag and the logical block n or range of logical block numbers to receiving the IO transaction, to tag manager 72. Tag table manager 72, in response, creates a new entry in tag table 76. Tag table manager 72 then stores into the newly created entry of table 76 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 62 is written to the physical memory block or blocks in disk d0 allocated to, according to CMM0, the logical block n or range of logical blocks beginning with block n of mirror M0. It is noted that the foregoing process is also employed by tag table manager 82. More particularly, presume storage manager 64 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M1. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 64 provides the tag and the logical block n or range of logical block numbers to receiving the IO transaction, to tag manager 82. Tag table manager 82, in response, creates a new entry in tag table 86. Tag table manager 82 then stores into the newly created entry of table 86 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 64 is written to the physical memory block or blocks in disk d1 allocated to, according to CMM1, the logical block n or range of logical blocks beginning with block n of mirror M1.

[0032] Tag tables 76 and 86 can be used to determine whether mirrors M0 and M1 are in synchronization. If the data processing system 40 operates perfectly, e.g., there is no crash of host 42 or interruption of power to disk arrays 44 and 46, then all IO transactions to write data to mirrors M1 and M0 should reach disk arrays 46 and 44, respectively, and the contents of tag tables 76 and 86 should be identical. However, if due to, for example, crash of host 42 one of two separate but related IO transactions do not reach disk array 44 or 46, then the contents of tag tables 76 and 86 should differ and mirrors M0 and M1 will be out of synchronization. To illustrate, suppose host 42 generates first and second IO transactions for writing data D_{new} to logical block 30 in mirrors M0 and M1. Further, presume that each of these separate IO transactions includes a unique tag 51 generated by generator 54. Lastly, presume that disk array 44 receives the first IO transaction from host 42, but disk array 46 does not receive the second IO transaction. As a result, tag table manager 72 creates a new entry m in tag table 76 which includes tag 51 and block number 30 while no such corresponding tag entry is created in tag table 86. Tag table 76 indicates that logical block 30 of mirror M0 has been updated with data D_{new}. Since table 86 does not include a corresponding entry, it can be said that logical block 30 in mirror M1 has not been updated with data D_{new}, and mirrors M0 and M1 are out of synchronization.

[0033] When mirrors M0 and M1 are out of synchronization, table 76 and 86 can be used to bring mirrors M1 and M0 back into synchronization. Mirrors M0 and M1 will be out of synchronization when an entry is found within table 76 that is not matched in table 86, or when an entry is found in table 86 that is not matched in table 76. When this happens, data is copied from mirror M1 to mirror M0 or from mirror M0 to mirror M1 according to the table entry that lacks the matching entry. To illustrate, tables 76 and 86 are not identical because table 76 includes an entry m consisting of a tag number 51 and logical block 30, and this entry cannot be found within table 86. To bring these mirrors M0 and M1 back into synchronization, data is copied from logical block 30 of mirror M0 to logical block 30 of mirror M1. Thereafter, entry m is deleted from tag table 76. Once this copying process is complete, and assuming there are no other mismatching entries between tables 76 and 86, mirrors M1 and M0 are back in synchronization.

[0034] In addition to creating or deleting entries in tag tables 76 and 86, tag table managers 72 and 82 can operate to (1) identify instances where mirrors M0 and M1 are out of synchronization or (2) delete matching entries in tag tables 76 and 86. Figure 6 illustrates these operational aspects of tag table manager 72. The operational aspects shown in Figure 6 can be initiated at any time. For example, the process shown in Figure 6 can be initiated after failure of host 42.

[0035] After the process is started in step 90, tag table manager 72 selects an entry in tag table 76 as shown in step 94 and then sends a message to tag table manager 82 asking if there is a corresponding match within tag table 86. The message sent to tag table manager 82 identifies the tag and logical block number or range of logical block numbers of the tag selected in step 92. Tag table manager 82 accesses tag table 86 using tag and logical block number or range of logical block numbers of the entry selected in step 92 that is provided in the message sent by tag table manager 72, and determines if a match exists as shown in step 94. If tag table manager 82 finds an entry in table 86 that matches the entry selected in table 76 (i.e., an entry in table 86 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in step 92), then tag table manager deletes the matching entry in table 86 and sends a reply message to tag table manager 72 indicating that table 86 did include a match to the entry selected in step 92. Tag manager 72, in response to receiving the message from tag table manager 82 that a matching entry did exist, deletes the entry selected in step 92. The entry deletion by tag managers 72 and 82 is shown in step 96. However, if tag table 86 does not have a match to the entry selected in table 76, then a message to that effect is sent back to tag table manager 72 and the process proceeds to step 100. In step 100, data is copied from mirror M0 to mirror M1. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M0 that are identified by the entry selected in step 92, is copied to mirror M1 at the logical memory block or range of logical memory blocks identified by the entry selected in step 92. For example, assuming the selected entry in step 92 is entry m of table 76, the data contents of logical block 30 of mirror M0 is copied to logical block 30 of mirror M1 in step 100. After copying data from mirror M0 to mirror M1in step 100, tag table manager 72 deletes the entry selected in table 76 as shown in step 104. Thereafter, as shown in step 104, tag table manager 72 checks tag table 76 to see if any tag entries remain. If an entry remains, then it is selected in step 92, and steps 94-102 are repeated. If no additional entries remain in table 76 at step 104, the process ends.

[0036] It is noted that tag table manager 82 operates under a process similar to that shown in Figure 6. In particular, tag table manager 82 selects an entry in tag table 86 and sends a message to tag table manager 72 asking if there is a corresponding match within tag table 76. The message sent to tag table manager 72 identifies the tag and logical block number or range of logical block numbers of the tag selected in table 86. Tag table manager 72 accesses tag table 76 using tag and logical block number or range of logical block numbers of the selected entry of table 86 provided in the message sent by tag table manager 82, and determines if a match exists in table 76. If tag table manager 72 finds an entry in table 76 that matches the entry selected in table 86 (i.e., an entry in table 76 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in table 86), then tag table manager 72 deletes the matching entry in table 76 and sends a reply message to tag table manager 82 indicating that table 76 did include a match to the entry selected in table 86. Tag manager 82, in response to receiving the message from tag table manager 72 that a matching entry did exist in table 76, deletes the selected entry in table 86. However, if tag table 76 does not have a match to the entry selected in table 86, then a message to that effect is sent to tag table manager 82 and data is copied from mirror M1 to mirror M0. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M1 that are identified by the entry selected in table 86 is copied to the logical memory block or range of logical memory blocks identified by the entry selected in table 86 but contained in mirror M0. After copying data from mirror M1 to mirror M0, tag table manager 82 deletes the selected entry in table 86. Thereafter, tag table manager 82 checks tag table 86 to see if any tag entries remain. If an entry remains, then it is selected, and the foregoing steps are repeated. If no additional entries remain in table 86, the process ends.

[0037] As noted above, storage subsystems may take form in OSDs. In this embodiment, tag tables, such as tag tables 76 and 86, my have to be modified to include an object number and information indicating whether the corresponding write operation is overwriting existing data of the storage object or whether the corresponding write operation extends the file length.

[0038] Although the present invention has been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein. On the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the scope of the invention as defined by the appended claims.

WE CLAIM:

1. A method comprising:

a computer system generating first and second write transactions;

wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;

the computer system transmitting the first and second write transactions to first and second storage devices, respectively.

2. The method of claim 1 further comprising:

the computer system generating third and fourth write transactions;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;

the computer system transmitting the third and fourth write transactions to the first and second storage devices, respectively.

3. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a logical block of a first storage object;

the second write transaction comprises data D to be written to a logical block of a second storage object.

4. The method of claim 3 further comprising:

the first storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory;

the second storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

5. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

6. The method of claim 5 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an identity of the range of logical blocks of the first storage object where data D is to be written, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an identity of the range of logical blocks in the second storage object where data D is to be written, wherein the second tag table is stored in second memory.

ATTORNEY/CLIENT PRIVILEGED AND CONFIDENTIAL

7. The method of claim 4 further comprising comparing the contents of one entry in the first tag table with the contents of entries in the second tag table to determine whether the second tag table includes an entry that matches the one entry.

8. The method of claim 7 further comprising copying data from physical memory allocated to a logical block number identified in the one entry to physical memory allocated to the second storage object if the second table lacks an entry with contents matching the contents of the one entry.

9. The method of claim 7 further comprising deleting the one entry in the first table if the second table contains an entry with contents that match the contents of the one entry.

10. The method of claim 9 further comprising deleting the entry in the second table with contents that match the contents of the one entry.

11. The method of claim 1 wherein the first tag is identical to the second tag.

12. The method of claim 1 further comprising:

the computer system generating a write transaction to write data to a logical block of a data volume;

the computer system incrementing a counter in response to generating the write transaction;

the computer system generating the first and second tags, wherein each of the first and second tags relate to the first and second write transactions, respectively, wherein the first and second tags are generated in response to generation of the write transaction, and wherein the first and second tags are generated as a function of an output of the incremented counter,.

13. The method of claim 1 wherein the first and second storage devices comprise first and second object storage devices.

14. The method of claim 11 wherein:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

17. The method of claim 16 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an indication that data D is to be stored in the extension of the first storage object, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an indication that data D is to be stored in the extension of the second storage object, wherein the second tag table is stored in second memory.

18. A method comprising:

a computer system generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

the computer system transmitting the transaction to a storage device.

19. The method of claim 18 wherein the storage device stores a replication of the storage object.

ATTORNEY/CLIENT PRIVILEGED AND CONFIDENTIAL

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIV E VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

ABSTRACT OF THE DISCLOSURE

[0039] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

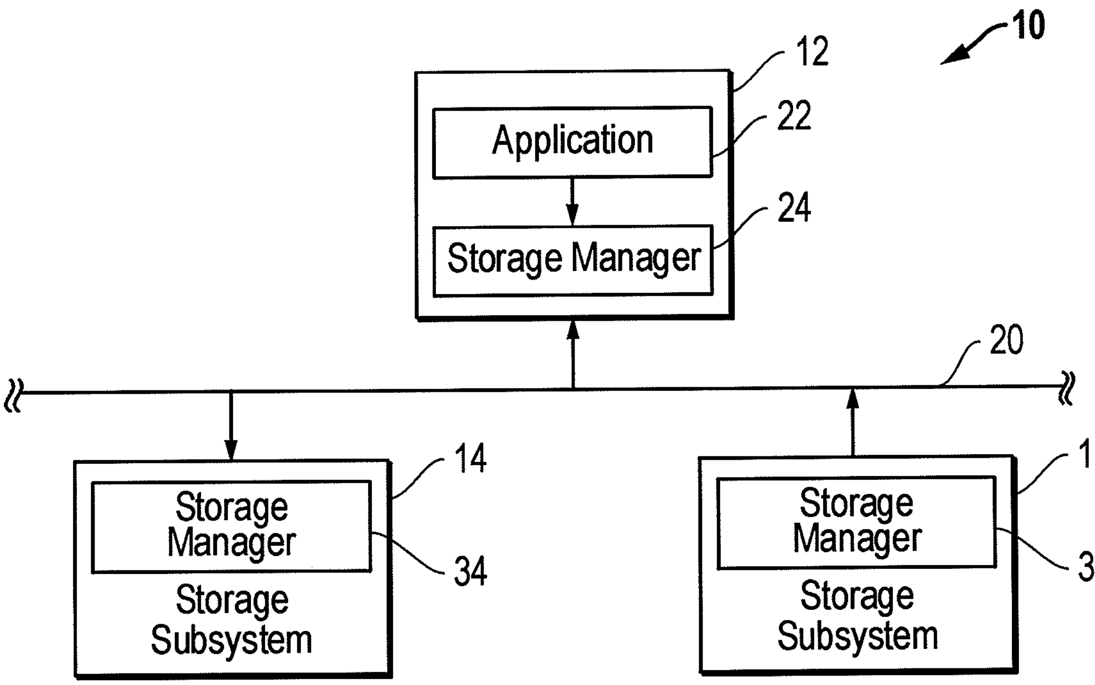

10
12
Application
22
Storage Manager
24
20
Storage
Manager
Storage
Subsystem
14
34
Storage
Manager
Storage
Subsystem
16
36

FIG. 1

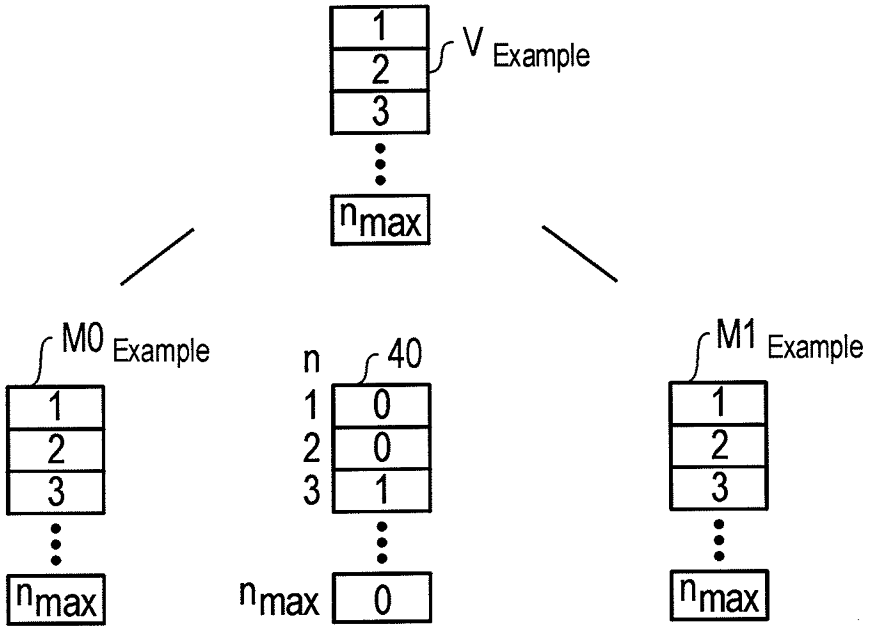

1
2
3
nmax
V Example
M0 Example
1
2
3
nmax
n
40
1 0
2 0
3 1
nmax 0
M1 Example
1
2
3
nmax

FIG. 2

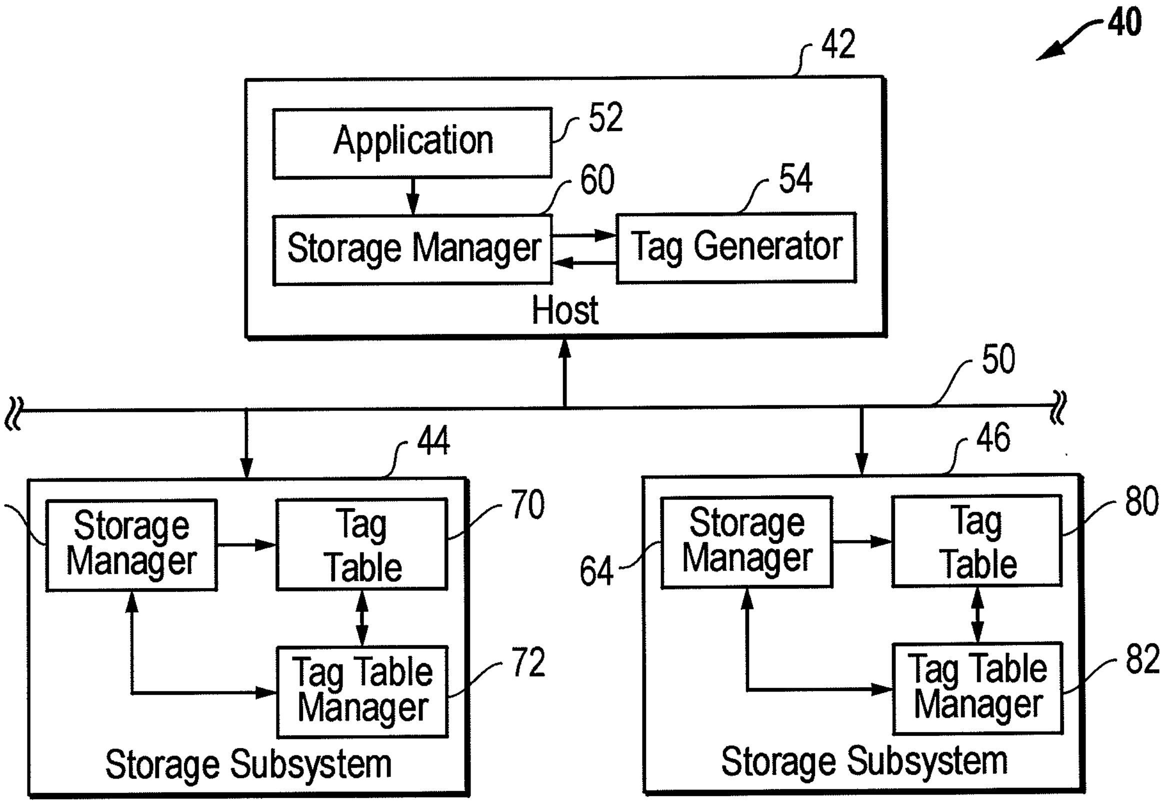

40
42
Application
52
60
54
Storage Manager
Tag Generator
Host
50
44
46
62
Storage Manager
Tag Table
70
80
64
Storage Manager
Tag Table
Tag Table Manager
72
Tag Table Manager
82
Storage Subsystem
Storage Subsystem

FIG. 3

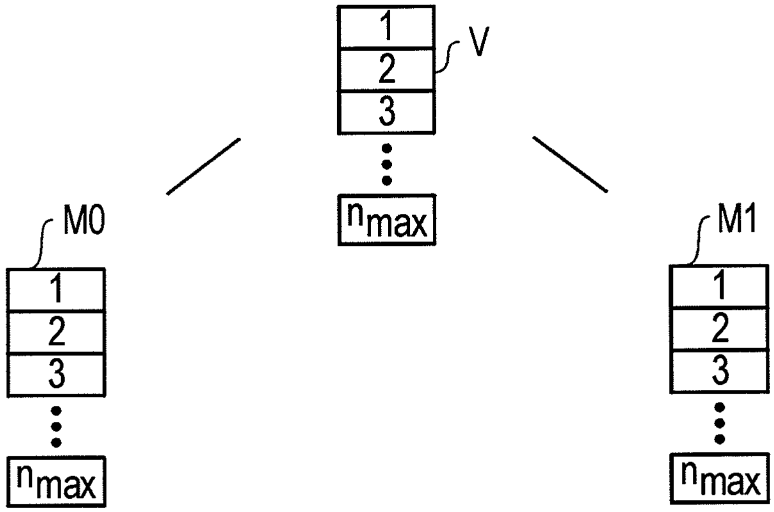

1
2
3
V
n_{max}
M0
1
2
3
n_{max}
M1
1
2
3
n_{max}

FIG. 4

76

	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
	⋮	⋮
m-1	17	2
m	51	30

86

	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
	⋮	⋮
m-1	17	2

FIG. 5

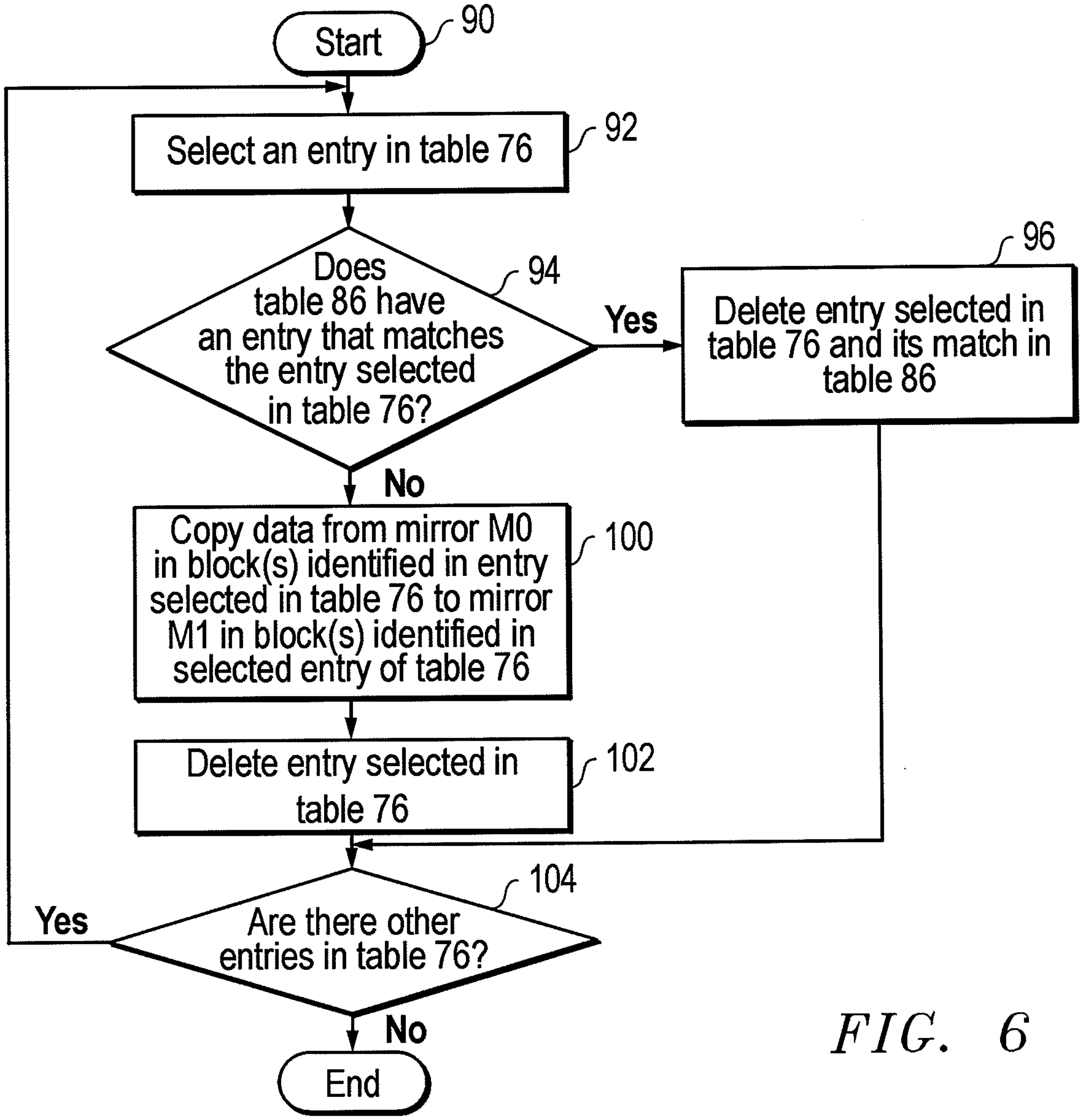

Start
90
Select an entry in table 76
92
Does table 86 have an entry that matches the entry selected in table 76?
94
Yes
Delete entry selected in table 76 and its match in table 86
96
No
Copy data from mirror M0 in block(s) identified in entry selected in table 76 to mirror M1 in block(s) identified in selected entry of table 76
100
Delete entry selected in table 76
102
Yes
Are there other entries in table 76?
104
No
End

FIG. 6

EXHIBIT F

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:55 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Second Draft Patent Application
Attachments: VRT0120US-Patent Application.DOC; VRT0120US-Dwgs fr Mario 2-20-04.pdf

From: Eric Stephenson
Sent: Monday, February 23, 2004 6:56 AM
To: Eric Stephenson; 'Ron Karr'; 'Narasimha Valiveti'; 'Ramana Jonnala (ramana@veritas.com)'; 'Dhanesh Joshi (dhanesh@veritas.com)'
Cc: 'julie.stephenson@veritas.com'; Roz Donaldson; Anne Castle; Sam Campbell
Subject: RE: VRTS 0655 (VRT0120US) Second Draft Patent Application

Sorry, I forgot to attach the second draft in my previous email.

Eric

> -----Original Message-----
> **From:** Eric Stephenson
> **Sent:** Monday, February 23, 2004 5:37 AM
> **To:** 'Ron Karr'; 'Narasimha Valiveti'; 'Ramana Jonnala (ramana@veritas.com)'; 'Dhanesh Joshi (dhanesh@veritas.com)'
> **Cc:** 'julie.stephenson@veritas.com'; Roz Donaldson; Anne Castle; Sam Campbell
> **Subject:** VRTS 0655 (VRT0120US) Second Draft Patent Application
>
> Greetings,
>
> Attached please find a second draft of the above identified patent application. This draft includes changes to the claims requested by Ron Karr.
>
> Please review the patent application including the claims. After your review, I will make any changes to the application you deem necessary.
>
> Please note that we seek to file this application by Thursday, February 26.
> Thanks,
>
> Eric
>
> Eric Stephenson
> Campbell Stephenson Ascolese LLP
> 4807 Spicewood Springs Road
> Suite 4-201
> Austin, Texas 78759
> (512) 439-5093 Direct
> (512) 439-5099 Fax
> estephenson@csapatent.com
>
> THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR THE PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED ABOVE. This message and the information contained herein is a confidential, attorney-client privileged communication. If you are not the

intended recipient, you have received this document in error, and any review, dissemination, distribution, or copying of this message is strictly prohibited. If you are an unintended recipient, please notify the sender immediately and delete this message and any copies thereof. Thank you.

EXHIBIT G

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:55 PM
To: Ronald Liu
Subject: FW: FW: VRTS 0655 (VRT0120US) Second Draft Patent Application

-----Original Message-----
From: Narasimha Valiveti [mailto:vnr@veritas.com]
Sent: Monday, February 23, 2004 9:24 PM
To: Eric Stephenson
Cc: tron@veritas.com; narasimha.valiveti@veritas.com; ramana@veritas.com; dhanesh@veritas.com; julie.stephenson@veritas.com
Subject: Re: FW: VRTS 0655 (VRT0120US) Second Draft Patent Application

Hi Eric,

Few comments on the second draft.

- I think the claim numbers got missed. Claims 15 and 16 are missing and 17 is referencing the claim 16 (this should be 14).

- I am lost with Claim 19 (replication of the storage object ??), are we saying that a replica storage object shall also stores the unique tag along with data it receives.

On page 12, in paragraph 0029, small typo, it should be "access to application 52."

A general comment : Its not explicitly stated (and i don't know whether we should or not) that in another embodiment the 'Tag Generator' can also be part of storage subsystem and it is the same storage subsystem that would be generating the first and second write transacations along with write tags.

Thanks for your help.

thanks,
- vnr

On Feb 23 Eric Stephenson (estephenson@csapatent.com) Wrote:

> Second attempt to deliver second draft.
>
> Please acknowledge receipt.
>
> Thanks,
>
> Eric
> -----Original Message-----
> From: Eric Stephenson
> Sent: Monday, February 23, 2004 6:56 AM
> To: Eric Stephenson; 'Ron Karr'; 'Narasimha Valiveti'; 'Ramana Jonnala
> (ramana@veritas.com)'; 'Dhanesh Joshi (dhanesh@veritas.com)'
> Cc: 'julie.stephenson@veritas.com'; Roz Donaldson; Anne Castle; Sam
> Campbell
> Subject: RE: VRTS 0655 (VRT0120US) Second Draft Patent Application
>
>
> Sorry, I forgot to attach the second draft in my previous email.
>
> Eric

>
> -----Original Message-----
> From: Eric Stephenson
> Sent: Monday, February 23, 2004 5:37 AM
> To: 'Ron Karr'; 'Narasimha Valiveti'; 'Ramana Jonnala
> (ramana@veritas.com)'; 'Dhanesh Joshi (dhanesh@veritas.com)'
> Cc: 'julie.stephenson@veritas.com'; Roz Donaldson; Anne Castle; Sam
> Campbell
> Subject: VRTS 0655 (VRT0120US) Second Draft Patent Application
>
>
>
>
> Greetings,
>
>
> Attached please find a second draft of the above identified patent
> application. This draft includes changes to the claims requested by
> Ron Karr.
>
> Please review the patent application including the claims. After your
> review, I will make any changes to the application you deem necessary.
>
> Please note that we seek to file this application by Thursday,
> February 26.
>
>
> Thanks,
>
> Eric
>
> Eric Stephenson
> Campbell Stephenson Ascolese LLP
> 4807 Spicewood Springs Road
> Suite 4-201
> Austin, Texas 78759
> (512) 439-5093 Direct
> (512) 439-5099 Fax
> estephenson@csapatent.com
>
>
>

EXHIBIT H

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:55 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Second Draft Patent Application

From: Dhanesh Joshi [mailto:Dhanesh@veritas.com]
Sent: Tuesday, February 24, 2004 1:12 AM
To: Eric Stephenson; Ron Karr; Narasimha Valiveti; Ramana Jonnala
Cc: Julie Stephenson
Subject: RE: VRTS 0655 (VRT0120US) Second Draft Patent Application

I have some questions/suggestions -

In addition to the algorithms described in section [0035] and [0036] I feel we need to include one more algorithm for clearing tags and detecting the missing tags. In this algorithm host (or tag generator subsystem) is responsible of clearing the tags. High level description is as follows -

To illustrate, presume storage manager 62 receives an IO transaction to write data to logical block n or a range of logical blocks beginning with block n of mirror M0. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 62 provides the tag and the logical block n or range of logical block numbers to receiving the IO transaction, to tag manager 72. Tag table manager 72, in response, creates a new entry in tag table 76. Tag table manager 72 then stores into the newly created entry of table 76 the received tag and logical block n or range of logical block numbers beginning with logical block n. When 62 returns status to the host 42, it also returns some token associated with the newly generated tag entry to the host 42. Storage manger 64 (inside 46) will behave the same way and it will return a token created for the tag entry in the table 80. When host receives status of IO from both 44 and 46, it declares IO as complete and returns status to the application 52. Host 42 has received a token from the 44 and 46 for the tag entries created in tag tables 70 and 80 respectively. At this point mirrors M0 and M1 are in consistent state and theoretically 44 and 46 can clear the tag from tables 70 and 80. When host 42 starts new IO transaction for 44 and 46, it will pass (piggyback) the previously received token from 44 and 46 to the respective system. Alternatively host can choose a certain time interval to send a message consisting of all accumulated tokens to a storage subsystem to clear the tags. Storage manger 62 and 64 on receiving such tokens, will identify the tag and then clear tag entries from the tag table.

If host 42 crashes after sending write to 44 but before sending same write to 46, that will leave M0 and M1 in inconsistent state. Before host 42 is restarted, synchronization authority will order 44 and 46 to bring M0 and M1 in a consistent state as described by sections 0033, 0035 and 0036.

Section [0035] and [0036] describes algorithms to determine if any of the tag is missing in any storage system 44 and 46. Mirrors M0 and M1 will be brought into synchronization after copying data from M0->M1 or from M1->M0, when any missing tag entry has been discovered. While doing such copy, it is obvious that, some kind of synchronization against further IOs from host 42 is required. For example consider a case where host 42 crashes after sending write to 44 but before sending

the same write to 46. If host 42 is restarted even before 44 and 46 can copy correct data between themselves, then IOs form 42 should be prevented till copy operation
has been completed. This process will need a some kind of master authority to instruct M0 and M1 (44 and 46) to synchronize IOs from 42 against the copy operation.

Do we have to explicitly mention such synchronization (and synchronization authority) against host IOs in the our application? OR high level description in section [0038] stating "this and other embodiments" is sufficient to cover this?

Again, I am not sure if we have to explicitly mention this algorithms or such things are already covered under this broader claim.

In section [0025] it says -
"Storage manager 62 is provided with a configuration map CMM0 which maps each logical block n of mirror M0 to a physical memory block x within hard disk d0. Lastly, storage manager 64 is provided with a configuration map CMM1 which maps each logical block n of mirror M1 to a physical memory block y within hard disk d1"

I think it should be "logical block of mirror M0 to a logical block X within hard disk d0". In my opinion wording like "physical memory block", used to describe disk blocks is not appropriate. This is true everywhere in the rest of the document where wording "physical memory block within disk" is used to describe disk blocks.

thanks
-- dhanesh

EXHIBIT I

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:56 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Second Draft Patent Application

From: Dhanesh Joshi [mailto:Dhanesh@veritas.com]
Sent: Wednesday, February 25, 2004 2:25 PM
To: Eric Stephenson; Ron Karr; Narasimha Valiveti; Ramana Jonnala
Cc: Julie Stephenson
Subject: RE: VRTS 0655 (VRT0120US) Second Draft Patent Application

Hi Eric,

> However, suppose storage manager 62 doesn't receive the new IO transaction with the piggybacked token as the res
> of some failure, but storage manager 64 does. If that happens, won't the tag tables 76 and 86 be out of sync thus
> indicating that mirrors M0 and M1 are out of sync?

[Dhanesh Joshi]
Yes that is correct. In this case M0 and M1 will be reported that they are out of sync even though they are not necessarily out of sync. Under such situation synchronization authority will Sync up some regions (or order 44 and 46 to sync up the regions) reported by the tag, even though some of them are not really required. It is indeed an overhead.. but it is ok to have such overhead during a recovery operation. Overall this algorithm helps to keep other part of the implementation very simple.

> Would it be acceptable to say that "logical block of mirror M0 is mapped to a disk block within hard disk d0?" If so, what is the difference
> between a physical memory block and a disk block?

[Dhanesh Joshi]
I feel it is okay to say "logical block of mirror M0 is mapped to a disk block within hard disk d0"
"physical memory block" typically means system's RAM. I feel it does not sound right when it is used with disk block.

Hope this helps

thanks
-- dhanesh

EXHIBIT J

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:56 PM
To: Ronald Liu
Subject: FW: FW: VRTS 0655 (VRT0120US) Second Draft Patent Application

-----Original Message-----
From: Narasimha Valiveti [mailto:vnr@veritas.com]
~~Sent: Wednesday, February 25, 2004 3:05 PM~~
To: Eric Stephenson
Cc: Narasimha Valiveti; tron@veritas.com; narasimha.valiveti@veritas.com; ramana@veritas.com; dhanesh@veritas.com; julie.stephenson@veritas.com
Subject: Re: FW: VRTS 0655 (VRT0120US) Second Draft Patent Application

On Feb 25 Eric Stephenson (estephenson@csapatent.com) Wrote:

> Thanks Narashima,
>
> I have added the following paragraph to the specification.
>
> It is noted that tag generator, in an alternative embodiment, could be placed in a different device of system 40. For example, tag generator 54 could be a part of one of the disk arrays 44 or 46. For purposes of explanation, it will be presumed that the tag generator is placed in host 42 as shown in Figure 4, unless otherwise noted.
>
> However, I'm a bit confused as to how or why storage subsystems would be generation the first and second write transactions. Please advise.
>
> Thanks Again.
>
> Eric

Hi Eric,

In an alternative embodiment, the host could forward the I/O transacation to one of storage subsystem with an instruction to generate the first and second write transactions (We call this a write forwarding request from the host). In such a scenario the tag generator could be part of the storage subsystem that is generating the first and second write transacations.

thanks,
- vnr

EXHIBIT K

ATTORNEY/CLIENT PRIVILEGED AND CONFIDENTIAL

THIS DRAFT PATENT APPLICATION CONTAINS INFORMATION CONFIDENTIAL TO *VERITAS Software Corporation* AND INCLUDES PRIVILEGED, ATTORNEY-CLIENT COMMUNICATIONS

EXPRESS MAIL LABEL NO.:

(EV 304738055 US)

DRAFT

8/10/2009; 3:10:14 PM

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

BACKGROUND OF THE INVENTION

[0001] Large scale data processing systems typically include several data storage subsystems (e.g., disk arrays) each containing many physical storage devices (e.g., hard disks) for storing critical data. Data processing systems often employ storage management systems to aggregate these physical storage devices to create highly reliable data storage. There are many types of highly reliable storage. Mirrored volume storage is an example. Mirrored volume storage replicates data over two or more mirrors of equal size. A logical memory block n of a mirrored volume maps to the same logical memory block n of each mirror. In turn, each logical memory block n of the mirrors map directly or indirectly to one or more disk blocks of one or more physical devices. Mirrored volumes provide data redundancy. If an application is unable to access data of one, a duplicate of the data sought should be available in an alternate mirror.

[0002] Figure 1 illustrates relevant components of an exemplary data processing system 10 that employs a two-way mirrored volume. While the present invention will be described with reference to a two-way mirrored volume, the present invention should not be limited thereto. The present invention may find use with other types of redundant storage including, for example, a three-way mirrored storage volume. Data processing system 10 includes a host (e.g., server computer system) 12 coupled to data storage subsystems 14 and 16 via storage interconnect 20. For purposes of explanation, storage interconnect 20 will take form in a storage area network (SAN) it being understood that the term storage interconnect should not be limited thereto. SAN 20 may include devices (e.g., switches, routers, hubs, etc.) that cooperate to transmit input/output (IO) transactions between host 12 and storage subsystems 14 and 16.

[0003] Each of the data storage subsystems 14 and 16 includes several physical storage devices. For purposes of explanation, data storage subsystems 14 and 16 are assumed to include several hard disks. The term physical storage device should not be limited to hard disks. Data storage subsystems 14 and 16 may take different forms. For example, data storage subsystem 14 may consist of "Just a Bunch of Disks" (JBOD) connected to an array controller card. Data storage subsystem 16 may consist of a block server appliance. For purposes of explanation, each of the data storage subsystems 14 and 16 will take form in an intelligent disk array, it being understood that the term data storage subsystem should not be limited thereto.

[0004] As noted, each of the disk arrays 14 and 16 includes several hard disks. The hard disk is the most popular permanent storage device currently used. A hard disk's total storage capacity is divided into many small chunks called physical memory blocks or disk blocks. For example, a 10 GB hard disk contains 20 million disk blocks, with each block able to hold 512 bytes of data. Any random disk block can be written to or read from in about the same time, without first having to read or write other disk blocks. Once written, a disk block continues to hold data even after the hard disk is powered down. While hard disks in general are reliable, they are subject to occasional failure. Data systems employ data redundancy schemes such as mirrored volumes to protect against occasional failure of a hard disk.

ATTORNEY/CLIENT PRIVILEGED AND CONFIDENTIAL

[0005] Host 12 includes an application 22 executing on one or more processors. Application 22 generates IO transactions to access critical data in response to receiving requests from client computer systems (not shown) coupled to host 12. In addition to application 22, host 12 includes a storage manager 24 executing on one or more processors. Volume Manager^{TM} provided by VERITAS Software Corporation of Mountain View, California, is an exemplary storage manager. Although many of the examples described herein will emphasize virtualization architecture and terminology associated with the VERITAS Volume Manager™, the software and techniques described herein can be used with a variety of different storage managers and architectures.

[0006] Storage managers can perform several functions. More particularly, storage managers can create storage objects (also known as virtualized disks) by aggregating hard disks such as those of disk arrays 14 and 16, underlying storage objects, or both. A storage object is an abstraction. Figure 2 shows a visual representation of exemplary storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ created for use in data processing system 10. Each of the storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ in Figure 2 consists of an array of n_{max} logical memory blocks that store or are configured to store data. While it is said that a logical memory block stores or is configured to store data, in reality the data is stored in one or more disk blocks of hard disks allocated directly or indirectly to the logical memory block.

[0007] Storage objects aggregated from hard disks can themselves be aggregated to form storage objects called logical data volumes. Logical data volumes are typically presented for direct or indirect use by an application such as application 22 executing on host 12. Thus, application 22 generates IO transactions to read data from or write data to one or more logical memory blocks of a data volume not knowing that the data volume is an aggregation of underlying storage objects, which in turn are aggregations of hard disks. Properties of storage objects depend on how the underlying storage objects or hard disks are aggregated. In other words, the method of aggregation determines the storage object type. In theory, there are a large number of possible methods of aggregation. The more common forms of aggregation include concatenated storage, striped storage, RAID storage, or mirrored storage. A more thorough discussion of how storage objects or hard disks can be aggregated can be found within Dilip M. Ranade [2002], "Shared Data Clusters" Wiley Publishing, Inc., which is incorporated herein by reference in its entirety.

[0008] Mirrored volumes provide highly reliable access to critical data. $V_{Example}$ of Figure 2 is an exemplary two-way mirrored volume. $V_{Example}$ was created by aggregating underlying storage objects (hereinafter mirrors) $M0_{Example}$ and $M1_{Example}$. Mirrors $M0_{Example}$ and $M1_{Example}$ were created by concatenating disk blocks from hard disks $d0_{Example}$ and $d1_{Example}$ (not shown) in disk arrays 14 and 16, respectively.

[0009] Storage managers can create storage object descriptions that describe the relationship between storage objects and their underlying storage objects or hard disks. These storage object descriptions typically include configuration maps. It is noted that storage object descriptions may include other information such as information indicating that a storage object is a snapshot copy of another storage object.

[0010] A configuration map maps a logical memory block of a corresponding storage object to one or more logical memory blocks of one or more underlying storage objects or to one or more disk blocks of one or more hard disks. To illustrate, configuration maps $CMV_{Example}$, $CMM0_{Example}$, and $CMM1_{Example}$ are created for mirrored volume $V_{Example}$ and underlying mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Configuration map $CMV_{Example}$ maps each logical memory block n of $V_{Example}$ to logical memory blocks n of mirrors $M0_{Example}$ and $M1_{Example}$. Configuration map $CMM0_{Example}$ maps each logical memory block n of mirror $M0_{Example}$ to a disk block x in hard disk $d0_{Example}$, while configuration map $CMM1_{Example}$ maps each logical memory block n of mirror $M1_{Example}$ to a disk block y in hard disk $d1_{Example}$. Configuration map $CMV_{Example}$ can be provided for use by storage manager 24, while configuration maps $CMM0_{Example}$ and $CMM1_{Example}$ can be provided for use by storage managers 34 and 36 executing on one or more processors in disk arrays 14 and 16, respectively.

[0011] Storage managers use configuration maps to translate IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks. To illustrate, presume an IO transaction is generated by application 22 to write data D to logical memory block 3 of data volume $V_{Example}$. This IO transaction is received directly or indirectly by storage manager 24. In turn, storage manager 24 accesses configuration map $CMV_{Example}$ and learns that logical memory block 3 is mapped to logical block 3 in both mirrors $M0_{Example}$ and $M1_{Example}$. It is noted storage manager may not receive the exact IO transaction generated by application 22. However, the transaction storage manager 24 receives will indicate that data D is to be written to block 3 of mirrored volume $V_{Example}$.

[0012] Storage manager 24 then generates first and second IO transactions to write data D to logical blocks 3 and mirrors $M0_{Example}$ and $M1_{Example}$, respectively. The IO transactions generated by storage manager 24 are transmitted to disk arrays 14 and 16, respectively, via SAN 20. Storage managers 34 and 36 of disk arrays 14 and 16, respectively receive directly or indirectly, the IO transactions sent by storage manager 24. It is noted that the IO transactions received by storage managers 34 and 36 may not be the exact IO transactions generated and sent by storage manager 24. Nonetheless, storage managers 34 and 36 will each receive an IO transaction to write data D to logical block 3 in mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Storage manager 34, in response to receiving the IO transaction, accesses configuration map $CMM0_{Example}$ to learn that block 3 of mirror $M0_{Example}$ is mapped to, for example, disk block 200 within hard disk d0. In response, a transaction is generated to write data D to disk block 200 within disk d0. Storage manager 36 accesses configuration map $CMM1_{Example}$ to learn that logical block 3 of mirror $M1_{Example}$ is mapped to, for example, disk block 300 within hard disk d1. In response, an IO transaction is generated for writing data D to disk block 300 within hard disk d1.

[0013] As noted above, while hard disks are reliable, hard disks are subject to failure. Data may be inaccessible within a failed hard disk. For this reason and others, administrators create mirrored data volumes. Unfortunately, data within mirrors of a mirrored volume may get into an inconsistent state if, for example, there is a server crash, storage power failure, or other problem which prevents data from being properly written to a hard disk. Consider the exemplary mirrored volume $V_{Example}$. Presume storage manager 24 generates first and second

IO transactions to write data D to logical blocks 3 in mirrors $M0_{Example}$ and $M1_{Example}$ in response to the IO transaction generated by application 22. Further, presume host 12 may fail after the first IO transaction is transmitted to disk array 14, but before the second IO transaction is transmitted to disk array 16. As a result, data D is written to disk block 200 within disk d0 of disk array 14, but data D is not written to disk block 300 within hard disk d1. When host 12 is restarted and exemplary volume $V_{Example}$ is made available again to application 22, mirrors $MO_{Example}$ and $M1_{Example}$ are said to be out of sync. In other words, mirrors $M0_{Example}$ and $M1_{Example}$ are no longer identical since at least block 3 in each contain different data. An IO transaction to read from logical memory block 3 of mirrored volume $V_{Example}$ could return either old or new data depending on whether the data is read from disk block 200 of hard disk d0 or disk block 300 of hard disk d1. Mirrors $MO_{Example}$ and $M1_{Example}$ should be resychronized before either is accessed again.

[0014] A brute force method to resynchronize mirrors $M0_{Example}$ and $M1_{Example}$ is simply to presume that one mirror (e.g., $M0_{Example}$) contains correct data and copy the contents of the one mirror to the other (e.g., $M1_{Example}$). It can take hours to resynchronize using this method. A smarter resynchronization technique is possible, but it requires some preparation. This alternate technique involves using what is called a dirty region map. Figure 2 illustrates a visual representation of an exemplary dirty region map 40 consisting of n_{max} entries corresponding to the n_{max} logical memory blocks within mirrors $M0_{Example}$ and $M1_{Example}$. Each entry of the dirty region map 40 indicates whether the corresponding blocks in mirrors are considered synchronized. For example, if entry n is set to logical 1, then blocks n in the mirrors are considered out of synchronization, and if n is set to logical 0, blocks n in the mirrors are considered in synchronization. Entry n in dirty region map 40 is set to logical 1 when the application 22 generates a transaction for writing data to logical block n of volume $V_{Example}$. Entry n is maintained in the logical 1 state until acknowledgement is received from both disk arrays 14 and 16 that data D has been written successfully to the disk blocks allocated to logical memory block n. However, if, using the example above, the first IO transaction to write data D to block 3 of mirror $M0_{Example}$ succeeds while the second IO transaction to write to block 3 of $M1_{Example}$ fails, then entry 3 and dirty region map 40 will be maintained as a logical 1 indicating that logical blocks 3 in the mirrors are out of synchronization. Mirrors $M0_{Example}$ and $M1_{Example}$ can be resynchronized by copying data from $M0_{Example}$ to mirror $M1_{Example}$, but for only logical memory blocks corresponding to dirty region map entries set to logical 1.

SUMMARY OF THE INVENTION

[0015] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

Fig. 1 illustrates a block diagram of a data system in which a logical data volume may be created;

Fig. 2 illustrates a visual representation of the storage objects created in the data system of Fig. 1;

Fig. 3, illustrates a block diagram of a data system capable of implementing one embodiment of the present invention;

Figs. 4 shows visual representations of a mirrored data volume employed in the data system of Fig. 3;

Fig. 5 is a visual representation of exemplary tag tables employed in the disk arrays of Fig. 3;

Fig.6 is a flow chart illustrating relevant aspects of managing the tag tables shown in Fig. 5.

[0017] The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION

[0018] Figure 3 illustrates in block diagram form, relevant components of an exemplary data processing system 40 capable of employing one embodiment of the present invention. Data processing system 40 includes a host (e.g., server computer system) 42 coupled to data storage subsystems 44 an 46 via storage interconnect 50. The present invention should not be limited to use in a data processing system consisting of only two storage subsystems. The present invention may be employed in a data processing system consisting of more than two storage subsystems.

[0019] Storage interconnect 50 takes form in a SAN consisting of several devices (e.g., switches, routers, etc.) that operate to transmit data between devices 42-46 including IO transactions to read or write data, it being understood that the term storage interconnect should not be limited thereto. SAN 50 can operate according to any one of a number of different protocols including Fibre Channel, Ethernet, etc. Each of the devices 42-46 includes one or more processors capable of processing data according to instructions of a software component such as a storage manager. As such, each of the devices 42-46 can be considered a computer system.

[0020] Data storage subsystems 44 and 46 may take different forms. For example, data storage subsystems 44 and 46 may take form in object storage devices (OSDs). Unless otherwise noted, each of the data storage subsystems 44 and 46 is implemented as a disk array, it being understood that the term data storage subsystem should not be limited thereto. Each of the disk arrays 44 and 46 includes several hard disks. Moreover, each of the disk arrays 44 and 46 may include a memory for storing a tag table that will be more fully described below.

[0021] Host 42 includes an application 52 executing on one or more processors. Application 52 may take one of many different forms. For example, application 52 may take form in a database management system (DBMS), customer relationship management software, etc. Application 52 generates IO transactions to read data from or write data to a data volume in response to receiving requests from client computer systems (not shown) coupled to host 42. Host 42 also includes a tag generator 54 executing on one or more processors. It is noted that tag generator, in an alternative embodiment, could be placed in a different device of system 40. For example, tag generator 54 could be a part of one of the disk arrays 44 or 46. For purposes of explanation, it will be presumed that the tag generator is placed in host 42 as shown in Figure 4, unless otherwise noted. Tag generator 54 will be more fully described below.

[0022] Devices 42-46 include a storage manager 60-64, respectively, executing on one or more processors. Each of the storage managers 60-64 is capable of performing many functions. More particularly, each of the storage managers 60-64 is capable of translating IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks of disk arrays 44 and 46. Storage managers 60-64 translate IO transactions according to configuration maps provided thereto. For purposes of illustration, it will be presumed that application 52 is provided with access to a mirrored data volume V having two mirrors, storage objects M0 and M1. It is noted that the present invention can be employed with respect to a data volume consisting of more than two mirrors.

[0023] A visual representation of volume V and mirrors M0 and M1 is found within Figure 4. As shown in Figure 4, each of V, M0, and M1 consist of an array of n_{max} logical memory blocks. It is noted that mirrors M0 and M1 could be formed from underlying storage objects (e.g., logical units or LUNs), or that mirrors M0 and M1 could be created as striped or RAID storage over hard disks or LUNs. It will be presumed, however, that M0 is formed by aggregating disk blocks contained within disk d0 (not shown) of disk array 44, and that mirror M1 is formed by aggregating disk blocks contained within disk d1 of disk array 46.

[0024] Storage managers 60-64 are provided with storage object descriptions for storage objects V, M0, and M1, respectively. Generally, the storage object descriptions define the relationships between storage objects and their underlying storage objects or hard disks. Moreover, the storage object descriptions may include other information. For example, the storage object descriptions for M0 and M1 indicate that M0 and M1 are mirrors or each other.

As such, storage manager 62 knows that a mirror of storage object M0 is contained within storage subsystem 46, and storage manager 64 knows that a mirror of storage object M1 is contained within storage subsystem 44.

[0025] Each of the storage object descriptions may also include a configuration map. A configuration map CMV is provided to storage manager 60 which maps each logical block n of volume V to corresponding blocks n in mirrors M0 and M1. Storage manager 62 is provided with a configuration map CMM0 which maps each logical block n of mirror M0 to a disk block x within hard disk d0. Lastly, storage manager 64 is provided with a configuration map CMM1 which maps each logical block n of mirror M1 to a disk block y within hard disk d1. Storage manager 60 is capable of translating IO transactions received from application 52 using configuration map CMV. Thus, when application 52 generates a transaction to write a data to block n of volume V, storage manager in response generates first and second transactions for writing a data D to blocks n in mirrors M0 and M1, respectively. Storage manager 62, in response to receiving an IO transaction to write data to block n of mirror M0, uses its configuration map CMM0 to identify the corresponding disk block x within disk d0 to which data D is to be written. Likewise, storage manager 64, in response to receiving an IO transaction to write data D to block n of mirror M1, uses its configuration map CMM1 to identify the corresponding disk block y within disk d1 to which data D is to be written.

[0026] IO transactions are received by storage manager 60. Tag generator 54, in response to storage manager 60 receiving an IO transaction to write data, generates a tag unique to the IO transaction to write data. This tag is provided to storage manager 60. In one embodiment, the tag may be the count value of a counter. In this embodiment, the count value is incremented by one each time storage manager 60 receives an IO transaction to write data.

[0027] Storage manager 60 accesses its configuration map CMV in response to receiving an IO transaction to write data to logical memory block n of volume V or a range of logical memory blocks beginning with block n. For purposed of explanation, it will be presumed that storage manager 60 receives IO transactions to write data to one logical block n of volume V unless otherwise noted. From configuration map CMV, storage manager 60 learns that logical block n of volume V is mapped to a logical blocks n in mirrors M1 and M0. Accordingly, storage manager 60 generates first and second IO transactions to write data D to block n of mirrors M0 and M1. Each of these first and second IO transactions also includes the unique tag generated by tag generator 54. The first and second IO transactions, including the unique tag, are transmitted by host 42 to disk arrays 44 and 46, respectively, via SAN 50. It is noted that the protocol employed with SAN 50 may have to be modified to accommodate IO transactions that include a unique tag.

[0028] Storage manager 62 is in data communication with nonvolatile memory 70. Likewise, storage manager 64 is in data communication with nonvolatile memory 80. Nonvolatile memories 70 and 80 are configured to store tag tables, such as tag tables 76 and 86, respectively, shown within Figure 5. Tag tables 76 and 86 are accessible by tag table managers 72 and 82, respectively. As will be more fully described below, tag table managers 72 and 82 perform several functions including the creation or deletion entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 may take form in software instructions executing on one or more processors of disk arrays 44 and 46, respectively.

[0029] Tag tables 76 and 86 are associated with mirrors M0 and M1, respectively. It is noted that additional tag tables like tag table 76 or 86 may be contained in memories 70 and 80, where each additional tag table is associated with a mirror of a mirrored data volume provided for access to application 52. However, for purposes of explanation, it will be presumed that memories 70 and 80 store only tag tables 76 and 86, respectively.

[0030] Each tag table 76 and 86 includes a plurality of entries. Each entry stores a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number. Alternatively, each tag table entry stores a tag and a physical block number or a range of disk block numbers beginning with a first disk block number. For purposes of explanation, it will be presumed that each entry of the tag tables 76 and 86 includes a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number unless otherwise noted. To illustrate, entry 2 of tag table 76 includes a tag equal to 2 and a logical block number equal to 25, and entry 3 of tag table 76 includes a tag equal to 70 and a range of 4 logical blocks beginning with logical memory block number 29.

[0031] Tag tables 76 and 86 track write IO transactions received by storage managers 62 and 64, respectively. Entries in tag tables 76 and 86 are created by tag managers 72 and 82, respectively, each time storage managers 62 and 64, respectively, receive IO transactions to write data to mirrors M0 and M1, respectively. The contents (i.e., the tag and logical block number or range of logical block numbers) of the tag table entries are defined by the IO write transactions received by storage managers 62 and 64. To illustrate, presume storage manager 62 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M0. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 62 provides the tag and the logical block n or range of logical block numbers of the received IO transaction, to tag manager 72. Tag table manager 72, in response, creates a new entry in tag table 76. Tag table manager 72 then stores into the newly created entry of table 76 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 62 is written to the disk block or blocks in disk d0 allocated to, according to CMM0, the logical block n or range of logical blocks beginning with block n of mirror M0. It is noted that the foregoing process is also employed by tag table manager 82. More particularly, presume storage manager 64 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M1. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 64 provides the tag and the logical block n or range of logical block numbers to receiving the IO transaction, to tag manager 82. Tag table manager 82, in response, creates a new entry in tag table 86. Tag table manager 82 then stores into the newly created entry of table 86 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 64 is written to the disk block or blocks in disk d1 allocated to, according to CMM1, the logical block n or range of logical blocks beginning with block n of mirror M1.

[0032] Tag tables 76 and 86 can be used to determine whether mirrors M0 and M1 are in synchronization. If the data processing system 40 operates perfectly, e.g., there is no crash of host 42 or interruption of power to disk arrays 44 and 46, then all IO transactions to write data to mirrors M1 and M0 should reach disk arrays 46 and 44, respectively, and the contents of tag tables 76 and 86 should be identical. However, if due to, for example, crash of host 42 one of two separate but related IO transactions do not reach disk array 44 or 46, then the contents of tag tables 76 and 86 should differ and mirrors M0 and M1 will be out of synchronization. To illustrate, suppose host 42 generates first and second IO transactions for writing data D_{new} to logical block 30 in mirrors M0 and M1. Further, presume that each of these separate IO transactions includes a unique tag 51 generated by generator 54. Lastly, presume that disk array 44 receives the first IO transaction from host 42, but disk array 46 does not receive the second IO transaction. As a result, tag table manager 72 creates a new entry m in tag table 76 which includes tag 51 and block number 30 while no such corresponding tag entry is created in tag table 86. Tag table 76 indicates that logical block 30 of mirror M0 has been updated with data D_{new}. Since table 86 does not include a corresponding entry, it can be said that logical block 30 in mirror M1 has not been updated with data D_{new}, and mirrors M0 and M1 are out of synchronization.

[0033] In another embodiment, tokens can be generated by disk arrays 44 and 46 when IO transactions are received. To illustrate, presume storage managers 62 and 64 receive first and second IO transactions to write data to logical block n or a range of logical blocks beginning with block n of mirrors M0 and M1, respectively. Each of the first and second IO transactions include the same tag generated by tag generator 54 and which is unique to the first and second IO transactions. Storage manager 62 provides the tag and the logical block n or range of logical block numbers for the received first IO transaction, to tag manager 72. Likewise, storage manager 64 provides the tag and the logical block n or range of logical block numbers for the received second IO transaction, to tag manager 82. Tag table managers 72 and 82, in response, create new entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 then store into the newly created entries of tables 76 and 86 the tag and logical block n or range of logical block numbers beginning with logical block n. Thereafter each of tag managers 72 and 82 generate first and second tokens by, for example, concatenating the tags of the first and second IO transactions, respectively, with the same prefix. If and when storage managers 62 and 64 return status of the first and second IO transactions, respectively, to host 42, storage managers 62 and 64 also return the first and second tokens associated with the newly generated tag entry. The first and second tokens should be identical to each other. If host 42 receives status from both disk arrays 44 and 46, it may declare the first and second IO transactions as complete, and host 42 will receive the first and second tokens. At this point host 42 knows that mirrors M0 and M1 are consistent with each other, at least with respect to the first and second IO transactions, since host 42 receives the matching first and second tokens. Presume now that host 42 generates third and fourth IO transactions to write data to block m of mirrors M0 and M1, respectively. Because host 42 has received the matching first and second tokens, host 42 could add the first and second tokens to the third and fourth IO transactions before they are sent to disk arrays 44 and 46, respectively. Tag managers 72 and 82 are provided with the first and second tokens, respectively, and remove the prefix. Tag managers 72 and 82 then compare the tags that remain after removing the prefix to the tags of entries in tables 76 and 86, respectively. When matches are found in the tag tables, the corresponding entries are removed. In yet another alternative, host 42 can choose a certain time interval to send to disk arrays 44 and 46 a message consisting of all matching tokens host 42. The accumulated tokens are subsequently provided to tag table managers 72 and 82. The prefixes of the accumulated tokens are removed, and tag managers 72 and 82 then compare the tags that remain after removing the prefix to the tags of entries in tables 76 and 86, respectively. When matches are found in the tag tables, the corresponding entries are removed.

[0034] When mirrors M0 and M1 are out of synchronization, table 76 and 86 can be used to bring mirrors M1 and M0 back into synchronization. Mirrors M0 and M1 will be out of synchronization when an entry is found within table 76 that is not matched in table 86, or when an entry is found in table 86 that is not matched in table 76. When this happens, data is copied from mirror M1 to mirror M0 or from mirror M0 to mirror M1 according to the table entry that lacks the matching entry. To illustrate, tables 76 and 86 are not identical because table 76 includes an entry m consisting of a tag number 51 and logical block 30, and this entry cannot be found within table 86. To bring these mirrors M0 and M1 back into synchronization, data is copied from logical block 30 of mirror M0 to logical block 30 of mirror M1. Thereafter, entry m is deleted from tag table 76. Once this copying process is complete, and assuming there are no other mismatching entries between tables 76 and 86, mirrors M1 and M0 are back in synchronization.

[0035] In addition to creating or deleting entries in tag tables 76 and 86, tag table managers 72 and 82 can operate to (1) identify instances where mirrors M0 and M1 are out of synchronization or (2) delete matching entries in tag tables 76 and 86. Figure 6 illustrates these operational aspects of tag table manager 72. The operational aspects shown in Figure 6 can be initiated at any time. For example, the process shown in Figure 6 can be initiated after failure of host 42.

[0036] After the process is started in step 90, tag table manager 72 selects an entry in tag table 76 as shown in step 94 and then sends a message to tag table manager 82 asking if there is a corresponding match within tag table 86. The message sent to tag table manager 82 identifies the tag and logical block number or range of logical block numbers of the tag selected in step 92. Tag table manager 82 accesses tag table 86 using tag and logical block number or range of logical block numbers of the entry selected in step 92 that is provided in the message sent by tag table manager 72, and determines if a match exists as shown in step 94. If tag table manager 82 finds an entry in table 86 that matches the entry selected in table 76 (i.e., an entry in table 86 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in step 92), then tag table manager deletes the matching entry in table 86 and sends a reply message to tag table manager 72 indicating that table 86 did include a match to the entry selected in step 92. Tag manager 72, in response to receiving the message from tag table manager 82 that a matching entry did exist, deletes the entry selected in step 92. The entry deletion by tag managers 72 and 82 is shown in step 96. However, if tag table 86 does not have a match to the entry selected in table 76, then a message to that effect is sent back to tag table manager 72 and the process proceeds to step 100. In step 100, data is copied from mirror M0 to mirror M1. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M0 that are identified by the entry selected in step 92, is copied to mirror M1 at the logical memory block or range of logical memory blocks identified by the entry selected in step 92. For example, assuming the selected entry in step 92 is entry m of table 76, the data contents of logical block 30 of mirror M0 is copied to logical block 30 of mirror M1 in step 100. After copying data from mirror M0 to mirror M1in step 100, tag table manager 72 deletes the entry selected in table 76 as shown in step 104. Thereafter, as shown in step 104, tag table manager 72 checks tag table 76 to see if any tag entries remain. If an entry remains, then it is selected in step 92, and steps 94-102 are repeated. If no additional entries remain in table 76 at step 104, the process ends.

[0037] It is noted that tag table manager 82 operates under a process similar to that shown in Figure 6. In particular, tag table manager 82 selects an entry in tag table 86 and sends a message to tag table manager 72 asking if there is a corresponding match within tag table 76. The message sent to tag table manager 72 identifies the tag and logical block number or range of logical block numbers of the tag selected in table 86. Tag table manager 72 accesses tag table 76 using tag and logical block number or range of logical block numbers of the selected entry of table 86 provided in the message sent by tag table manager 82, and determines if a match exists in table 76. If tag table manager 72 finds an entry in table 76 that matches the entry selected in table 86 (i.e., an entry in table 76 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in table 86), then tag table manager 72 deletes the matching entry in table 76 and sends a reply message to tag table manager 82 indicating that table 76 did include a match to the entry selected in table 86. Tag manager 82, in response to receiving the message from tag table manager 72 that a matching entry did exist in table 76, deletes the selected entry in table 86. However, if tag table 76 does not have a match to the entry selected in table 86, then a message to that effect is sent to tag table manager 82 and data is copied from mirror M1 to mirror M0. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M1 that are identified by the entry selected in table 86 is copied to the logical memory block or range of logical memory blocks identified by the entry selected in table 86 but contained in mirror M0. After copying data from mirror M1 to mirror M0, tag table manager 82 deletes the selected entry in table 86. Thereafter, tag table manager 82 checks tag table 86 to see if any tag entries remain. If an entry remains, then it is selected, and the foregoing steps are repeated. If no additional entries remain in table 86, the process ends.

[0038] If host 42 crashes after sending a write IO transaction to disk array 44 but before sending the same IO transaction to disk array 46, that will leave M0 and M1 in inconsistent state. Before host 42 is restarted, synchronization authority will generate and send a message to disk arrays 44 and 46 instructing them to bring M0 and M1 into a consistent state as described by the foregoing sections.

[0039] While a resynchronization of mirrors M0 and M1 occur after, for example a crash of host 42, it may be necessary to suspend host 42 from further generation of IO transactions to write data to mirrors M0 and M1 until mirrors M0 and M1 are brought back into a consistent state.

[0040] As noted above, storage subsystems may take form in OSDs. In this embodiment, tag tables, such as tag tables 76 and 86, my have to be modified to include an object number and information indicating whether the corresponding write operation is overwriting existing data of the storage object or whether the corresponding write operation extends the file length.

[0041] It was noted above that in an alternative embodiment, the tag generator may be placed in one of the disk arrays 44 or 46. In this alternative embodiment, host 42 could forward the IO transaction from application 52 to disk array 44 (for example) that contains the tag generator with an instruction for disk array 44 to generate first and second write IO transactions that contain a unique tag. The first IO transaction would be provided to the storage manager 62 while the second IO transaction is transmitted to storage manager 64 in disk array 46.

[0042] Although the present invention has been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein. On the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the scope of the invention as defined by the appended claims.

WE CLAIM:

1. A method comprising:

a computer system generating first and second write transactions;

wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;

the computer system transmitting the first and second write transactions to first and second storage devices, respectively.

2. The method of claim 1 further comprising:

the computer system generating third and fourth write transactions;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;

the computer system transmitting the third and fourth write transactions to the first and second storage devices, respectively.

3. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a logical block of a first storage object;

the second write transaction comprises data D to be written to a logical block of a second storage object.

4. The method of claim 3 further comprising:

the first storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory;

the second storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

5. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

6. The method of claim 5 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an identity of the range of logical blocks of the first storage object where data D is to be written, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an identity of the range of logical blocks in the second storage object where data D is to be written, wherein the second tag table is stored in second memory.

7. The method of claim 4 further comprising comparing the contents of one entry in the first tag table with the contents of entries in the second tag table to determine whether the second tag table includes an entry that matches the one entry.

8. The method of claim 7 further comprising copying data from physical memory allocated to a logical block number identified in the one entry to physical memory allocated to the second storage object if the second table lacks an entry with contents matching the contents of the one entry.

9. The method of claim 7 further comprising deleting the one entry in the first table if the second table contains an entry with contents that match the contents of the one entry.

10. The method of claim 9 further comprising deleting the entry in the second table with contents that match the contents of the one entry.

11. The method of claim 1 further comprising:

the computer system generating a write transaction to write data to a logical block of a data volume;

the computer system incrementing a counter in response to generating the write transaction;

the computer system generating the first and second tags, wherein each of the first and second tags relate to the first and second write transactions, respectively, wherein the first and second tags are generated in response to generation of the write transaction, and wherein the first and second tags are generated as a function of an output of the incremented counter,.

12. The method of claim 1 wherein the first and second storage devices comprise first and second object storage devices.

13. The method of claim 1 wherein:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

14. The method of claim 13 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an indication that data D is to be stored in the extension of the first storage object, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an indication that data D is to be stored in the extension of the second storage object, wherein the second tag table is stored in second memory.

15. A method comprising:

a computer system generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

the computer system transmitting the transaction to a storage device.

16. The method of claim 16 wherein the storage device stores a replication of the storage object.

17. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

generating first and second write transactions;

wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;

transmitting the first and second write transactions directly or indirectly to first and second storage devices, respectively.

18. The computer readable medium of claim 17 wherein the method further comprises:

generating third and fourth write transactions;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;

transmitting the third and fourth write transactions directly or indirectly to the first and second storage devices, respectively.

19. The computer readable medium of claim 17 wherein:

the first write transaction comprises data D to be written to a logical block of a first storage object;

the second write transaction comprises data D to be written to a logical block of a second storage object.

20. The computer readable medium of claim 17 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

21. The computer readable medium of claim 17 wherein the first tag is identical to the second tag.

22. The computer readable medium of claim 17 wherein the method further comprises:

generating a write transaction to write data to a logical block of a data volume;

incrementing a counter in response to generating the write transaction;

generating the first and second tags, wherein each of the first and second tags relate to the first and second write transactions, respectively, wherein the first and second tags are generated in response to generation of the write transaction, and wherein the first and second tags are generated as a function of an output of the incremented counter,.

23. The computer readable medium of claim 17 the first and second storage devices comprise first and second object storage devices.

24. The computer readable medium of claim 17:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

25. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

transmitting the transaction to a storage device.

26. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

in response to receiving a first transaction comprising a first tag, storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory, wherein the first tag corresponds to a second tag of a second write transaction;

the second storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

7. The method of claim 4 further comprising comparing the contents of one entry in the first tag table with the contents of entries in the second tag table to determine whether the second tag table includes an entry that matches the one entry.

8. The method of claim 7 further comprising copying data from physical memory allocated to a logical block number identified in the one entry to physical memory allocated to the second storage object if the second table lacks an entry with contents matching the contents of the one entry.

9. The method of claim 7 further comprising deleting the one entry in the first table if the second table contains an entry with contents that match the contents of the one entry.

10. The method of claim 9 further comprising deleting the entry in the second table with contents that match the contents of the one entry.

ATTORNEY/CLIENT PRIVILEGED AND CONFIDENTIAL

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIV E VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

ABSTRACT OF THE DISCLOSURE

[0043] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

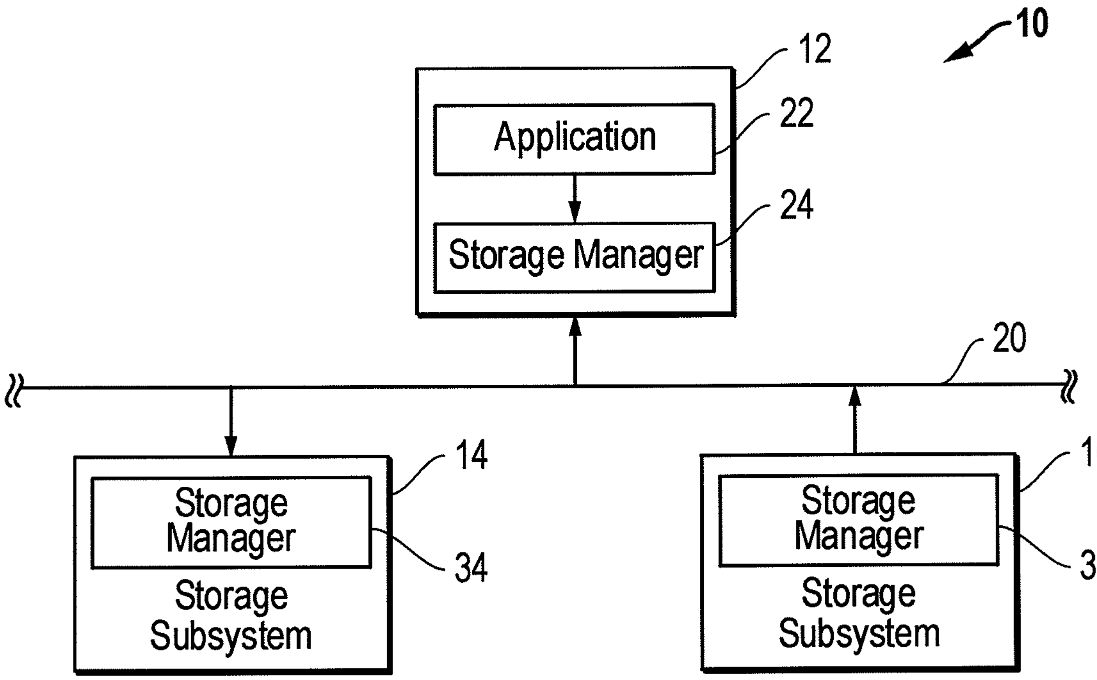

10
12
Application
22
Storage Manager
24
20
14
Storage Manager
34
Storage Subsystem
16
Storage Manager
36
Storage Subsystem

FIG. 1

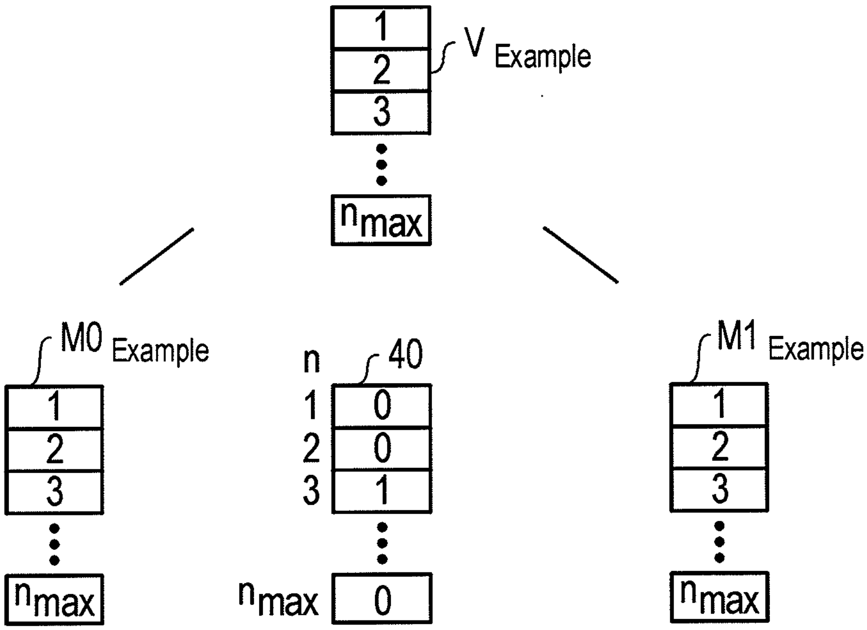

1
2
3
n_{max}
V Example
M0 Example
1
2
3
n_{max}
n
40
1 0
2 0
3 1
n_{max} 0
M1 Example
1
2
3
n_{max}

FIG. 2

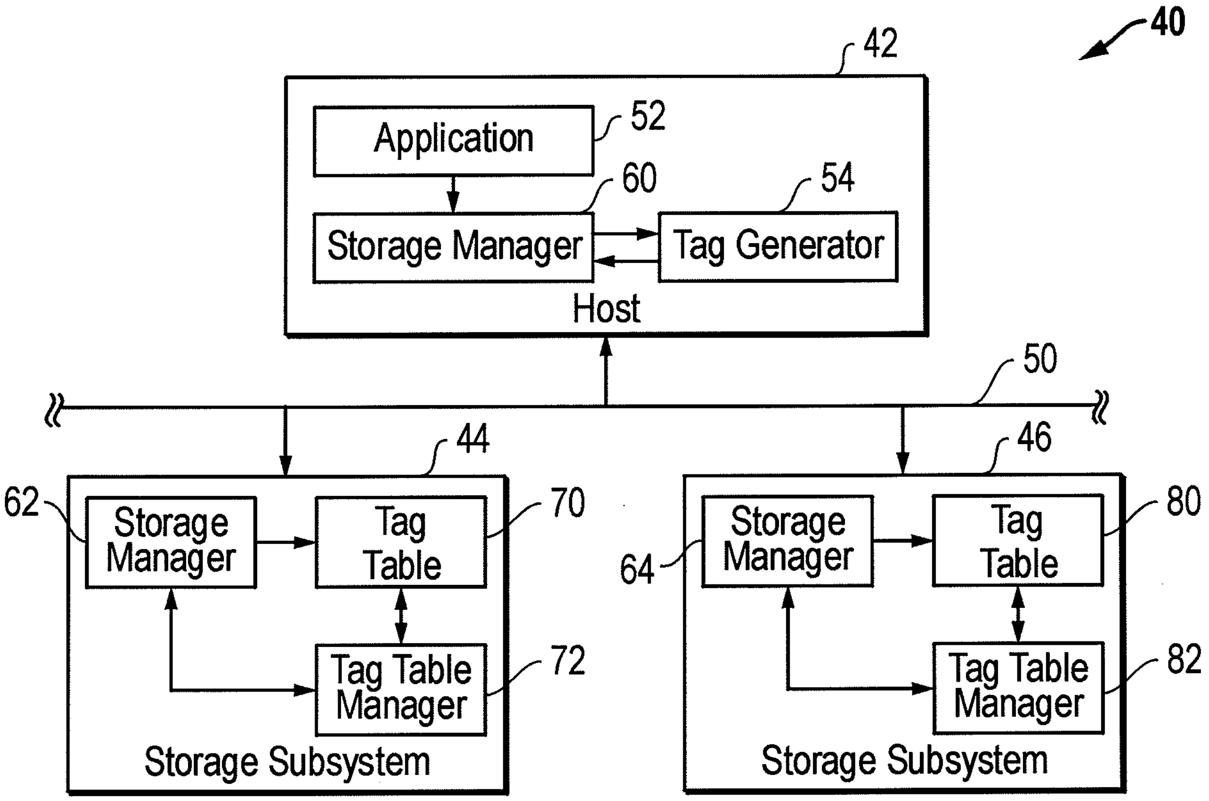

40
42
Application
52
60
54
Storage Manager
Tag Generator
Host
50
44
46
62
Storage Manager
Tag Table
70
Tag Table Manager
72
Storage Subsystem
64
Storage Manager
Tag Table
80
Tag Table Manager
82
Storage Subsystem

FIG. 3

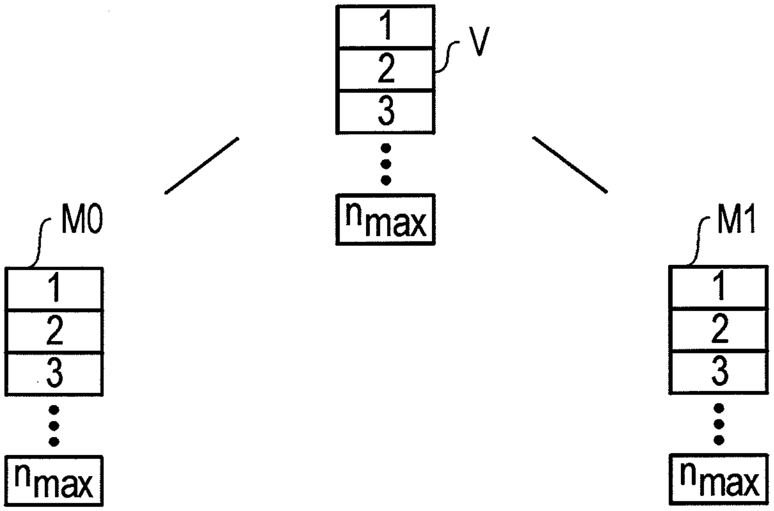
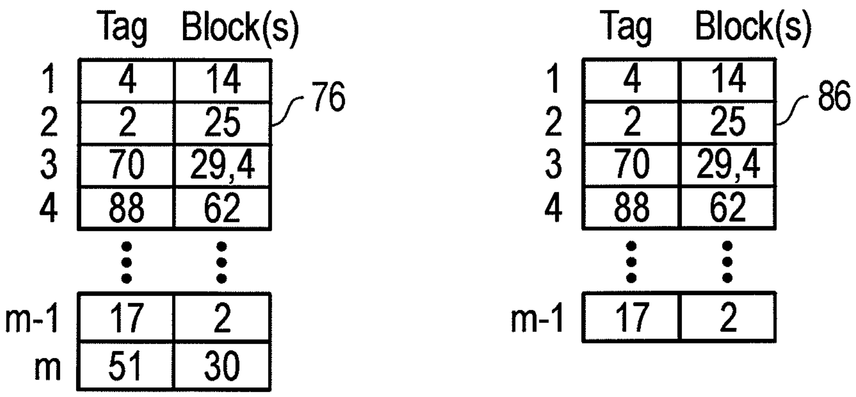

1
2
3
n_{max}
V
M0
1
2
3
n_{max}
M1
1
2
3
n_{max}

FIG. 4



	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
⋮	⋮	⋮
m-1	17	2
m	51	30

	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
⋮	⋮	⋮
m-1	17	2

FIG. 5

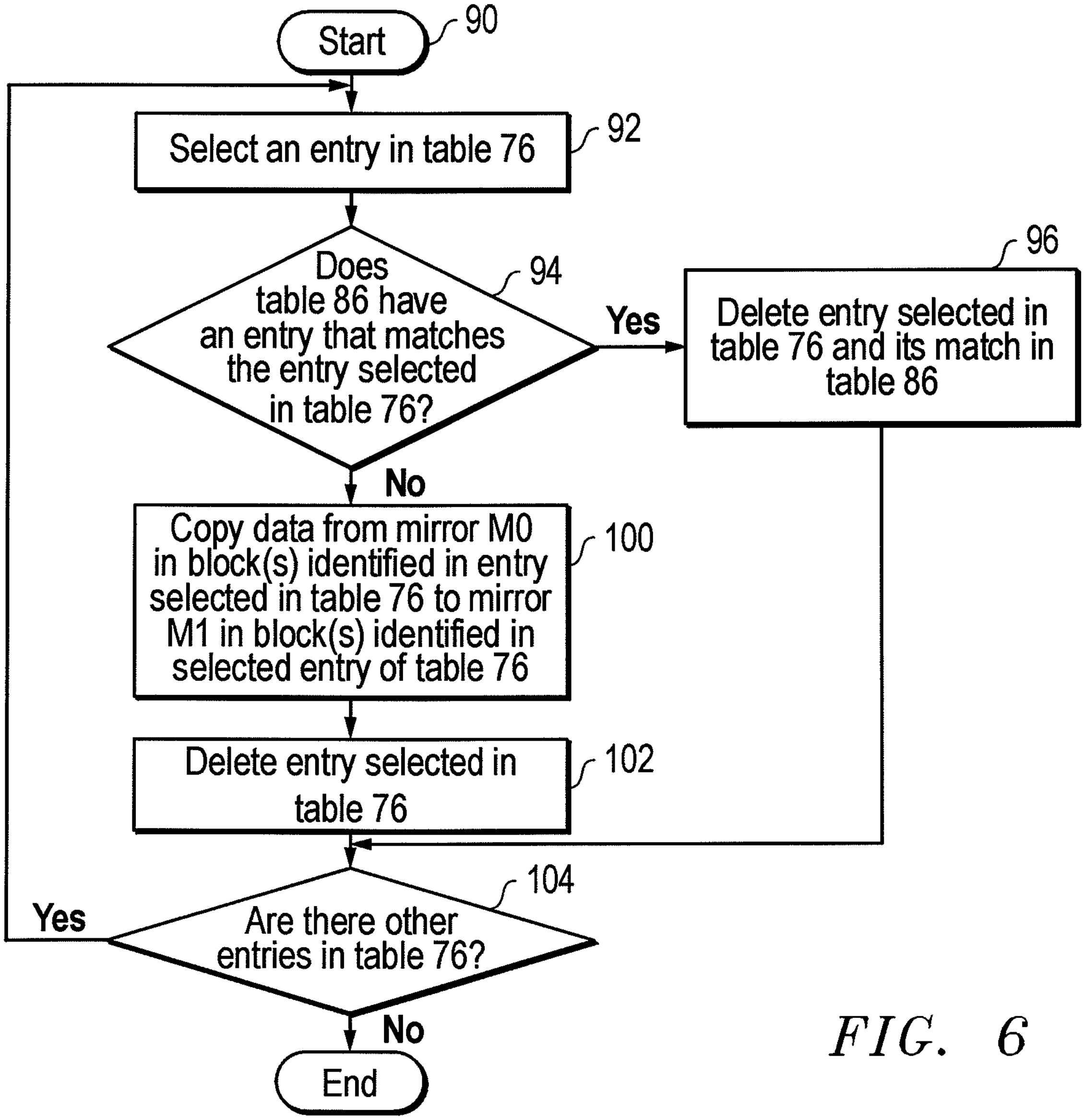

Start
90
Select an entry in table 76
92
Does table 86 have an entry that matches the entry selected in table 76?
94
Yes
Delete entry selected in table 76 and its match in table 86
96
No
Copy data from mirror M0 in block(s) identified in entry selected in table 76 to mirror M1 in block(s) identified in selected entry of table 76
100
Delete entry selected in table 76
102
Are there other entries in table 76?
104
Yes
No
End

FIG. 6

DECLARATION FOR PATENT APPLICATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below adjacent to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of subject matter (process, machine, manufacture, or composition of matter, or an improvement thereof) which is claimed and for which a patent is sought by way of the application entitled

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

which (check) ☒ is attached hereto.
□ and is amended by the Preliminary Amendment attached hereto.
□ was filed on ___ as Application Serial No. ____
□ and was amended on (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
Number	Country	Day/Month/Year Filed	Yes	No
			□	□

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

Provisional Application Number	Filing Date

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) or PCT international application(s) designating the United States of America listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal

Regulations, § 1.56, which became available between the filing date of the prior application(s) and the national or PCT international filing date of this application:

Application Serial No.	Filing Date	Status (patented, pending, abandoned)

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith:

Joseph FitzGerald (33,881); John Brigden (40,530); Julie Stephenson (41,330); and

Marc R. Ascolese (42,268); Brenna A. Brock (48,309); Sam G. Campbell (42,381); Justin M. Dillon (42,486); D'Ann Rifai (47,026); Eric A. Stephenson (38,321).

Please address all correspondence and telephone calls to:

Eric A. Stephenson
CAMPBELL STEPHENSON ASCOLESE LLP
4807 Spicewood Springs Road
Building 4, Suite 201
Austin, Texas 78759
Telephone: 512-439-5093
Facsimile: 512-439-5099

I declare that all statements made herein of my own knowledge are true, all statements made herein on information and belief are believed to be true, and all statements made herein are made with the knowledge that whoever, in any matter within the jurisdiction of the Patent and Trademark Office, knowingly and willfully falsifies, conceals, or covers up by any trick, scheme, or device a material fact, or makes any false, fictitious or fraudulent statements or representations, or makes or uses any false writing or document knowing the same to contain any false, fictitious or fraudulent statement or entry, shall be subject to the penalties including fine or imprisonment or both as set forth under 18 U.S.C. 1001, and that violations of this paragraph may jeopardize the validity of the application or this document, or the validity or enforceability of any patent, trademark registration, or certificate resulting therefrom.

Full name of sole (or first joint inventor): Ronald S. Karr

Inventor's Signature: ______________________ Date: ______________

Residence: Palo Alto, California 94301

Post Office Address: 333 Ramona Street
Palo Alto, California 94301

Citizenship: U.S.A.

Full name of second inventor: Ramana Jonnala
Inventor's Signature: ______________________ Date: ______________
Residence: Sunnyvale, California 94086
Post Office Address: 825 E. Evelyn Avenue, #326
Sunnyvale, California 94086
Citizenship: U.S.A.

Full name of third inventor: Narasimha R. Valiveti
Inventor's Signature: ______________________ Date: ______________
Residence: Sunnyvale, California 94085
Post Office Address: 395 Ano Nuevo Avenue, Apt #413
Sunnyvale, California 94085
Citizenship: India

Full name of fourth inventor: Dhanesh Joshi
Inventor's Signature: ______________________ Date: ______________
Residence: Santa Clara, California 95051
Post Office Address: 46, Cabot Avenue
Santa Clara, California 95051
Citizenship: India

EXHIBIT L

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:56 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Final Draft And Declaration
Attachments: VRT0120US-Patent Application.DOC; VRT0120US-Dwgs fr Mario 2-20-04.pdf; VRT0120US-Declaration.doc

---

From: Eric Stephenson
Sent: Wednesday, February 25, 2004 5:38 PM
To: Ron Karr; narasimha.valiveti@veritas.com; Ramana Jonnala; Dhanesh Joshi (dhanesh@veritas.com)
Cc: julie.stephenson@veritas.com; Roz Donaldson; Anne Castle
Subject: VRTS 0655 (VRT0120US) Final Draft And Declaration

Greetings,

Attached please find a final draft of the above identified patent application.

This final draft includes changes requested by Narasimha and Dhanesh.

Unless further changes are needed, please sign and fax to me the attached declaration. Please send the originally declaration to Julie Stephenson.

Very shortly after we file the application with the U.S. Patent Office, we will send you an assignment for your signature.

Thanks,

Eric

Eric Stephenson
Campbell Stephenson Ascolese LLP
4807 Spicewood Springs Road
Suite 4-201
Austin, Texas 78759
(512) 439-5093 Direct
(512) 439-5099 Fax
estephenson@csapatent.com

THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR THE PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED ABOVE. This message and the information contained herein is a confidential, attorney-client privileged communication. If you are not the intended recipient, you have received this document in error, and any review, dissemination, distribution, or copying of this message is strictly prohibited. If you are an unintended recipient, please notify the sender immediately and delete this message and any copies thereof. Thank you.

EXHIBIT M

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:56 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Final Draft And Declaration

From: Eric Stephenson
Sent: Wednesday, February 25, 2004 6:02 PM
To: Eric Stephenson; 'Ron Karr'; 'narasimha.valiveti@veritas.com'; 'Ramana Jonnala'; 'Dhanesh Joshi (dhanesh@veritas.com)'
Cc: julie.stephenson@veritas.com
Subject: RE: VRTS 0655 (VRT0120US) Final Draft And Declaration

FYI, please ignore the claims highlighted in yellow. I forgot to delete them.

Sorry about that.

Eric

> -----Original Message-----
> **From:** Eric Stephenson
> **Sent:** Wednesday, February 25, 2004 5:38 PM
> **To:** Ron Karr; narasimha.valiveti@veritas.com; Ramana Jonnala; Dhanesh Joshi (dhanesh@veritas.com)
> **Cc:** julie.stephenson@veritas.com; Roz Donaldson; Anne Castle
> **Subject:** VRTS 0655 (VRT0120US) Final Draft And Declaration
>
> Greetings,
>
> Attached please find a final draft of the above identified patent application.
>
> This final draft includes changes requested by Narasimha and Dhanesh.
>
> Unless further changes are needed, please sign and fax to me the attached declaration. Please send the originally declaration to Julie Stephenson.
>
> Very shortly after we file the application with the U.S. Patent Office, we will send you an assignment for your signature.
>
> Thanks,
>
> Eric
>
> Eric Stephenson
> Campbell Stephenson Ascolese LLP
> 4807 Spicewood Springs Road
> Suite 4-201
> Austin, Texas 78759
> (512) 439-5093 Direct
> (512) 439-5099 Fax
> estephenson@csapatent.com

THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR THE PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED ABOVE. This message and the information contained herein is a confidential, attorney-client privileged communication. If you are not the intended recipient, you have received this document in error, and any review, dissemination, distribution, or copying of this message is strictly prohibited. If you are an unintended recipient, please notify the sender immediately and delete this message and any copies thereof. Thank you.

EXHIBIT N

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:57 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Final Draft And Declaration

From: Eric Stephenson
Sent: Thursday, February 26, 2004 11:09 AM
To: Eric Stephenson; 'Ron Karr'; 'narasimha.valiveti@veritas.com'; 'Ramana Jonnala'; 'Dhanesh Joshi (dhanesh@veritas.com)'
Subject: RE: VRTS 0655 (VRT0120US) Final Draft And Declaration

Greetings,

Did you get a chance to sign the declaration?

Thanks,

Eric

> -----Original Message-----
> **From:** Eric Stephenson
> **Sent:** Wednesday, February 25, 2004 6:02 PM
> **To:** Eric Stephenson; 'Ron Karr'; 'narasimha.valiveti@veritas.com'; 'Ramana Jonnala'; 'Dhanesh Joshi (dhanesh@veritas.com)'
> **Cc:** julie.stephenson@veritas.com
> **Subject:** RE: VRTS 0655 (VRT0120US) Final Draft And Declaration
>
> FYI, please ignore the claims highlighted in yellow. I forgot to delete them.
>
> Sorry about that.
>
> Eric
>
>> -----Original Message-----
>> **From:** Eric Stephenson
>> **Sent:** Wednesday, February 25, 2004 5:38 PM
>> **To:** Ron Karr; narasimha.valiveti@veritas.com; Ramana Jonnala; Dhanesh Joshi (dhanesh@veritas.com)
>> **Cc:** julie.stephenson@veritas.com; Roz Donaldson; Anne Castle
>> **Subject:** VRTS 0655 (VRT0120US) Final Draft And Declaration
>>
>> Greetings,
>>
>> Attached please find a final draft of the above identified patent application.
>>
>> This final draft includes changes requested by Narasimha and Dhanesh.
>>
>> Unless further changes are needed, please sign and fax to me the attached declaration. Please send the originally declaration to Julie Stephenson.

Very shortly after we file the application with the U.S. Patent Office, we will send you an assignment for your signature.

Thanks,

Eric

Eric Stephenson
Campbell Stephenson Ascolese LLP
4807 Spicewood Springs Road
Suite 4-201
Austin, Texas 78759
(512) 439-5093 Direct
(512) 439-5099 Fax
estephenson@csapatent.com

THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR THE PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED ABOVE. This message and the information contained herein is a confidential, attorney-client privileged communication. If you are not the intended recipient, you have received this document in error, and any review, dissemination, distribution, or copying of this message is strictly prohibited. If you are an unintended recipient, please notify the sender immediately and delete this message and any copies thereof. Thank you.

EXHIBIT O

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:57 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Final Draft And Declaration

---

From: Dhanesh Joshi [mailto:Dhanesh@veritas.com]
~~**Sent:** Thursday, February 26, 2004 2:22 PM~~
To: Eric Stephenson; Ron Karr; Narasimha Valiveti; Ramana Jonnala
Subject: RE: VRTS 0655 (VRT0120US) Final Draft And Declaration

Hi Eric,
sorry for not being very clear in my previous mail. A small correction -

> Thereafter each of tag managers 72 and 82 generate first and second tokens by, for example, concatenating the tags
> of the first and second IO transactions

Each of the tag managers 72 and 82 generate the first and second token respectively. These token basically used to identify the tag entry in the tag table. These tokens can even be some simple numbers to identify the tag entry.

> The first and second tokens should be identical to each other.

This is not correct. As 72 and 82 may generate the tokens by any algorithm that they like, they may or may not be the same. This logic does not require them to be the same. No one, except the tag manager that generated a particular token, tries to interpret the token. i.e. 72 only understands/interprets the token that it generated.

> At this point host 42 knows that mirrors M0 and M1 are consistent with each other, at least with respect to the first
> and second IO transactions, since host 42 receives the matching first and second tokens.

Host 42 interprets the "result" of the transaction but it does not interpret the token. 42 does not care if they are matching or not. If the returned status from both, 72 and 82 is SUCCESS, then it knows that M0 and M1 are in sync.

> Presume now that host 42 generates third and fourth IO transactions to write data to block m of mirrors M0 and M1,
> respectively. Because host 42 has received the matching first and second tokens, host 42 could add the first and
> second tokens to the third and fourth IO transactions before they are sent to disk arrays 44 and 46, respectively.

A token is considered to be ready for "piggyback" only after all related IO transaction are completed. For example,
consider that host 42 starts 3rd and 4th transaction before both 1st and 2nd are completed. Lets consider

at this time that 1st is completed but second is not. So 42 has status and token of the 1st transaction. In this case it will not piggyback 1st token with the 3rd transaction.

> Tag managers 72 and 82 are provided with the first and second tokens, respectively, and remove the prefix.
> Tag managers 72 and 82 then compare the tags that remain after removing the prefix to the tags of entries in tables 76
> and 86, respectively. When matches are found in the tag tables, the corresponding entries are removed.

I did not understand what you meant by "remove prefix".

Tag managers 72 and 82 receives the piggybacked tokens. When they see such token they "know" that IO transaction
associated with that token has reached to the all "other" mirror as well and they can safely remove the tag entry in the table. 72 and 82 uses these tokens to "locate" the tag entry in the tag table.

thanks
-- dhanesh

EXHIBIT P

EXPRESS MAIL LABEL NO.:

(EV 304738055 US)

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

BACKGROUND OF THE INVENTION

[0001] Large scale data processing systems typically include several data storage subsystems (e.g., disk arrays) each containing many physical storage devices (e.g., hard disks) for storing critical data. Data processing systems often employ storage management systems to aggregate these physical storage devices to create highly reliable data storage. There are many types of highly reliable storage. Mirrored volume storage is an example. Mirrored volume storage replicates data over two or more mirrors of equal size. A logical memory block n of a mirrored volume maps to the same logical memory block n of each mirror. In turn, each logical memory block n of the mirrors map directly or indirectly to one or more disk blocks of one or more physical devices. Mirrored volumes provide data redundancy. If an application is unable to access data of one, a duplicate of the data sought should be available in an alternate mirror.

[0002] Figure 1 illustrates relevant components of an exemplary data processing system 10 that employs a two-way mirrored volume. While the present invention will be described with reference to a two-way mirrored volume, the present invention should not be limited thereto. The present invention may find use with other types of redundant storage including, for example, a three-way mirrored storage volume. Data processing system 10 includes a host (e.g., server computer system) 12 coupled to data storage subsystems 14 and 16 via storage interconnect 20. For purposes of explanation, storage interconnect 20 will take form in a storage area network (SAN) it being understood that the term storage interconnect should

not be limited thereto. SAN 20 may include devices (e.g., switches, routers, hubs, etc.) that cooperate to transmit input/output (IO) transactions between host 12 and storage subsystems 14 and 16.

[0003] Each of the data storage subsystems 14 and 16 includes several physical storage devices. For purposes of explanation, data storage subsystems 14 and 16 are assumed to include several hard disks. The term physical storage device should not be limited to hard disks. Data storage subsystems 14 and 16 may take different forms. For example, data storage subsystem 14 may consist of "Just a Bunch of Disks" (JBOD) connected to an array controller card. Data storage subsystem 16 may consist of a block server appliance. For purposes of explanation, each of the data storage subsystems 14 and 16 will take form in an intelligent disk array, it being understood that the term data storage subsystem should not be limited thereto.

[0004] As noted, each of the disk arrays 14 and 16 includes several hard disks. The hard disk is the most popular permanent storage device currently used. A hard disk's total storage capacity is divided into many small chunks called physical memory blocks or disk blocks. For example, a 10 GB hard disk contains 20 million disk blocks, with each block able to hold 512 bytes of data. Any random disk block can be written to or read from in about the same time, without first having to read or write other disk blocks. Once written, a disk block continues to hold data even after the hard disk is powered down. While hard disks in general are reliable, they are subject to occasional failure. Data systems employ data redundancy schemes such as mirrored volumes to protect against occasional failure of a hard disk.

[0005] Host 12 includes an application 22 executing on one or more processors. Application 22 generates IO transactions to access critical data in response to receiving requests from client computer systems (not shown) coupled to host 12. In addition to application 22, host 12 includes a storage manager 24 executing on one or more processors. Volume Manager^{TM} provided by VERITAS Software Corporation of Mountain View, California, is an exemplary storage manager. Although many of the examples described herein will emphasize virtualization architecture and terminology associated with the VERITAS Volume Manager™, the software and techniques described herein can be used with a variety of different storage managers and architectures.

[0006] Storage managers can perform several functions. More particularly, storage managers can create storage objects (also known as virtualized disks) by aggregating hard disks such as those of disk arrays 14 and 16, underlying storage objects, or both. A storage object is an abstraction. Figure 2 shows a visual representation of exemplary storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ created for use in data processing system 10. Each of the storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ in Figure 2 consists of an array of n_{max} logical memory blocks that store or are configured to store data. While it is said that a logical memory block stores or is configured to store data, in reality the data is stored in one or more disk blocks of hard disks allocated directly or indirectly to the logical memory block.

[0007] Storage objects aggregated from hard disks can themselves be aggregated to form storage objects called logical data volumes. Logical data volumes are typically presented for direct or indirect use by an application such as application 22 executing on host 12. Thus, application 22 generates IO transactions to read data from or write data to one or more logical memory blocks of a data volume not knowing that the data volume is an aggregation of underlying storage objects, which in turn are aggregations of hard disks. Properties of storage objects depend on how the underlying storage objects or hard disks are aggregated. In other words, the method of aggregation determines the storage object type. In theory, there are a large number of possible methods of aggregation. The more common forms of aggregation include concatenated storage, striped storage, RAID storage, or mirrored storage. A more thorough discussion of how storage objects or hard disks can be aggregated can be found within Dilip M. Ranade [2002], "Shared Data Clusters" Wiley Publishing, Inc., which is incorporated herein by reference in its entirety.

[0008] Mirrored volumes provide highly reliable access to critical data. $V_{Example}$ of Figure 2 is an exemplary two-way mirrored volume. $V_{Example}$ was created by aggregating underlying storage objects (hereinafter mirrors) $M0_{Example}$ and $M1_{Example}$. Mirrors $M0_{Example}$ and $M1_{Example}$ were created by concatenating disk blocks from hard disks $d0_{Example}$ and $d1_{Example}$ (not shown) in disk arrays 14 and 16, respectively.

[0009] Storage managers can create storage object descriptions that describe the relationship between storage objects and their underlying storage objects or hard disks. These storage object descriptions typically include configuration maps. It is noted that

storage object descriptions may include other information such as information indicating that a storage object is a snapshot copy of another storage object.

[0010] A configuration map maps a logical memory block of a corresponding storage object to one or more logical memory blocks of one or more underlying storage objects or to one or more disk blocks of one or more hard disks. To illustrate, configuration maps $CMV_{Example}$, $CMM0_{Example}$, and $CMM1_{Example}$ are created for mirrored volume $V_{Example}$ and underlying mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Configuration map $CMV_{Example}$ maps each logical memory block n of $V_{Example}$ to logical memory blocks n of mirrors $M0_{Example}$ and $M1_{Example}$. Configuration map $CMM0_{Example}$ maps each logical memory block n of mirror $M0_{Example}$ to a disk block x in hard disk $d0_{Example}$, while configuration map $CMM1_{Example}$ maps each logical memory block n of mirror $M1_{Example}$ to a disk block y in hard disk $d1_{Example}$. Configuration map $CMV_{Example}$ can be provided for use by storage manager 24, while configuration maps $CMM0_{Example}$ and $CMM1_{Example}$ can be provided for use by storage managers 34 and 36 executing on one or more processors in disk arrays 14 and 16, respectively.

[0011] Storage managers use configuration maps to translate IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks. To illustrate, presume an IO transaction is generated by application 22 to write data D to logical memory block 3 of data volume $V_{Example}$. This IO transaction is received directly or indirectly by storage manager 24. In turn, storage manager 24 accesses configuration map $CMV_{Example}$ and learns that logical memory block 3 is mapped to logical block 3 in both mirrors $M0_{Example}$ and $M1_{Example}$. It is noted storage manager may not receive the exact IO transaction generated by application 22. However, the transaction storage manager 24 receives will indicate that data D is to be written to block 3 of mirrored volume $V_{Example}$.

[0012] Storage manager 24 then generates first and second IO transactions to write data D to logical blocks 3 and mirrors $M0_{Example}$ and $M1_{Example}$, respectively. The IO transactions generated by storage manager 24 are transmitted to disk arrays 14 and 16, respectively, via SAN 20. Storage managers 34 and 36 of disk arrays 14 and 16, respectively receive directly or indirectly, the IO transactions sent by storage manager 24. It is noted that the IO

transactions received by storage managers 34 and 36 may not be the exact IO transactions generated and sent by storage manager 24. Nonetheless, storage managers 34 and 36 will each receive an IO transaction to write data D to logical block 3 in mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Storage manager 34, in response to receiving the IO transaction, accesses configuration map $CMM0_{Example}$ to learn that block 3 of mirror $M0_{Example}$ is mapped to, for example, disk block 200 within hard disk d0. In response, a transaction is generated to write data D to disk block 200 within disk d0. Storage manager 36 accesses configuration map $CMM1_{Example}$ to learn that logical block 3 of mirror $M1_{Example}$ is mapped to, for example, disk block 300 within hard disk d1. In response, an IO transaction is generated for writing data D to disk block 300 within hard disk d1.

[0013] As noted above, while hard disks are reliable, hard disks are subject to failure. Data may be inaccessible within a failed hard disk. For this reason and others, administrators create mirrored data volumes. Unfortunately, data within mirrors of a mirrored volume may get into an inconsistent state if, for example, there is a server crash, storage power failure, or other problem which prevents data from being properly written to a hard disk. Consider the exemplary mirrored volume $V_{Example}$. Presume storage manager 24 generates first and second IO transactions to write data D to logical blocks 3 in mirrors $M0_{Example}$ and $M1_{Example}$ in response to the IO transaction generated by application 22. Further, presume host 12 may fail after the first IO transaction is transmitted to disk array 14, but before the second IO transaction is transmitted to disk array 16. As a result, data D is written to disk block 200 within disk d0 of disk array 14, but data D is not written to disk block 300 within hard disk d1. When host 12 is restarted and exemplary volume $V_{Example}$ is made available again to application 22, mirrors $MO_{Example}$ and $M1_{Example}$ are said to be out of sync. In other words, mirrors $M0_{Example}$ and $M1_{Example}$ are no longer identical since at least block 3 in each contain different data. An IO transaction to read from logical memory block 3 of mirrored volume $V_{Example}$ could return either old or new data depending on whether the data is read from disk block 200 of hard disk d0 or disk block 300 of hard disk d1. Mirrors $MO_{Example}$ and $M1_{Example}$ should be resychronized before either is accessed again.

[0014] A brute force method to resynchronize mirrors $M0_{Example}$ and $M1_{Example}$ is simply to presume that one mirror (e.g., $M0_{Example}$) contains correct data and copy the contents of the one mirror to the other (e.g., $M1_{Example}$). It can take hours to resynchronize using this method.

A smarter resynchronization technique is possible, but it requires some preparation. This alternate technique involves using what is called a dirty region map. Figure 2 illustrates a visual representation of an exemplary dirty region map 40 consisting of n_{max} entries corresponding to the n_{max} logical memory blocks within mirrors $M0_{Example}$ and $M1_{Example}$. Each entry of the dirty region map 40 indicates whether the corresponding blocks in mirrors are considered synchronized. For example, if entry n is set to logical 1, then blocks n in the mirrors are considered out of synchronization, and if n is set to logical 0, blocks n in the mirrors are considered in synchronization. Entry n in dirty region map 40 is set to logical 1 when the application 22 generates a transaction for writing data to logical block n of volume $V_{Example}$. Entry n is maintained in the logical 1 state until acknowledgement is received from both disk arrays 14 and 16 that data D has been written successfully to the disk blocks allocated to logical memory block n. However, if, using the example above, the first IO transaction to write data D to block 3 of mirror $M0_{Example}$ succeeds while the second IO transaction to write to block 3 of $M1_{Example}$ fails, then entry 3 and dirty region map 40 will be maintained as a logical 1 indicating that logical blocks 3 in the mirrors are out of synchronization. Mirrors $M0_{Example}$ and $M1_{Example}$ can be resynchronized by copying data from $M0_{Example}$ to mirror $M1_{Example}$, but for only logical memory blocks corresponding to dirty region map entries set to logical 1.

SUMMARY OF THE INVENTION

[0015] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

Fig. 1 illustrates a block diagram of a data system in which a logical data volume may be created;

Fig. 2 illustrates a visual representation of the storage objects created in the data system of Fig. 1;

Fig. 3, illustrates a block diagram of a data system capable of implementing one embodiment of the present invention;

Figs. 4 shows visual representations of a mirrored data volume employed in the data system of Fig. 3;

Fig. 5 is a visual representation of exemplary tag tables employed in the disk arrays of Fig. 3;

Fig.6 is a flow chart illustrating relevant aspects of managing the tag tables shown in Fig. 5.

[0017] The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION

[0018] Figure 3 illustrates in block diagram form, relevant components of an exemplary data processing system 40 capable of employing one embodiment of the present invention. Data processing system 40 includes a host (e.g., server computer system) 42 coupled to data storage subsystems 44 an 46 via storage interconnect 50. The present invention should not be limited to use in a data processing system consisting of only two storage subsystems. The present invention may be employed in a data processing system consisting of more than two storage subsystems.

[0019] Storage interconnect 50 takes form in a SAN consisting of several devices (e.g., switches, routers, etc.) that operate to transmit data between devices 42-46 including IO transactions to read or write data, it being understood that the term storage interconnect should not be limited thereto. SAN 50 can operate according to any one of a number of different protocols including Fibre Channel, Ethernet, etc. Each of the devices 42-46 includes one or more processors capable of processing data according to instructions of a software component such as a storage manager. As such, each of the devices 42-46 can be considered a computer system.

[0020] Data storage subsystems 44 and 46 may take different forms. For example, data storage subsystems 44 and 46 may take form in object storage devices (OSDs). Unless otherwise noted, each of the data storage subsystems 44 and 46 is implemented as a disk array, it being understood that the term data storage subsystem should not be limited thereto. Each of the disk arrays 44 and 46 includes several hard disks. Moreover, each of the disk arrays 44 and 46 may include a memory for storing a tag table that will be more fully described below.

[0021] Host 42 includes an application 52 executing on one or more processors. Application 52 may take one of many different forms. For example, application 52 may take form in a database management system (DBMS), customer relationship management software, etc. Application 52 generates IO transactions to read data from or write data to a data volume in response to receiving requests from client computer systems (not shown) coupled to host 42. Host 42 also includes a tag generator 54 executing on one or more

processors. It is noted that tag generator, in an alternative embodiment, could be placed in a different device of system 40. For example, tag generator 54 could be a part of one of the disk arrays 44 or 46. For purposes of explanation, it will be presumed that the tag generator is placed in host 42 as shown in Figure 4, unless otherwise noted. Tag generator 54 will be more fully described below.

[0022] Devices 42-46 include a storage manager 60-64, respectively, executing on one or more processors. Each of the storage managers 60-64 is capable of performing many functions. More particularly, each of the storage managers 60-64 is capable of translating IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks of disk arrays 44 and 46. Storage managers 60-64 translate IO transactions according to configuration maps provided thereto. For purposes of illustration, it will be presumed that application 52 is provided with access to a mirrored data volume V having two mirrors, storage objects M0 and M1. It is noted that the present invention can be employed with respect to a data volume consisting of more than two mirrors.

[0023] A visual representation of volume V and mirrors M0 and M1 is found within Figure 4. As shown in Figure 4, each of V, M0, and M1 consist of an array of n_{max} logical memory blocks. It is noted that mirrors M0 and M1 could be formed from underlying storage objects (e.g., logical units or LUNs), or that mirrors M0 and M1 could be created as striped or RAID storage over hard disks or LUNs. It will be presumed, however, that M0 is formed by aggregating disk blocks contained within disk d0 (not shown) of disk array 44, and that mirror M1 is formed by aggregating disk blocks contained within disk d1 of disk array 46.

[0024] Storage managers 60-64 are provided with storage object descriptions for storage objects V, M0, and M1, respectively. Generally, the storage object descriptions define the relationships between storage objects and their underlying storage objects or hard disks. Moreover, the storage object descriptions may include other information. For example, the storage object descriptions for M0 and M1 indicate that M0 and M1 are mirrors or each other. As such, storage manager 62 knows that a mirror of storage object M0 is contained within storage subsystem 46, and storage manager 64 knows that a mirror of storage object M1 is contained within storage subsystem 44.

[0025] Each of the storage object descriptions may also include a configuration map. A configuration map CMV is provided to storage manager 60 which maps each logical block n of volume V to corresponding blocks n in mirrors M0 and M1. Storage manager 62 is provided with a configuration map CMM0 which maps each logical block n of mirror M0 to a disk block x within hard disk d0. Lastly, storage manager 64 is provided with a configuration map CMM1 which maps each logical block n of mirror M1 to a disk block y within hard disk d1. Storage manager 60 is capable of translating IO transactions received from application 52 using configuration map CMV. Thus, when application 52 generates a transaction to write a data to block n of volume V, storage manager in response generates first and second transactions for writing a data D to blocks n in mirrors M0 and M1, respectively. Storage manager 62, in response to receiving an IO transaction to write data to block n of mirror M0, uses its configuration map CMM0 to identify the corresponding disk block x within disk d0 to which data D is to be written. Likewise, storage manager 64, in response to receiving an IO transaction to write data D to block n of mirror M1, uses its configuration map CMM1 to identify the corresponding disk block y within disk d1 to which data D is to be written.

[0026] IO transactions are received by storage manager 60. Tag generator 54, in response to storage manager 60 receiving an IO transaction to write data, generates a tag unique to the IO transaction to write data. This tag is provided to storage manager 60. In one embodiment, the tag may be the count value of a counter. In this embodiment, the count value is incremented by one each time storage manager 60 receives an IO transaction to write data.

[0027] Storage manager 60 accesses its configuration map CMV in response to receiving an IO transaction to write data to logical memory block n of volume V or a range of logical memory blocks beginning with block n. For purposed of explanation, it will be presumed that storage manager 60 receives IO transactions to write data to one logical block n of volume V unless otherwise noted. From configuration map CMV, storage manager 60 learns that logical block n of volume V is mapped to a logical blocks n in mirrors M1 and M0. Accordingly, storage manager 60 generates first and second IO transactions to write data D to block n of mirrors M0 and M1. Each of these first and second IO transactions also includes the unique tag generated by tag generator 54. The first and second IO transactions, including

the unique tag, are transmitted by host 42 to disk arrays 44 and 46, respectively, via SAN 50. It is noted that the protocol employed with SAN 50 may have to be modified to accommodate IO transactions that include a unique tag.

[0028] Storage manager 62 is in data communication with nonvolatile memory 70. Likewise, storage manager 64 is in data communication with nonvolatile memory 80. Nonvolatile memories 70 and 80 are configured to store tag tables, such as tag tables 76 and 86, respectively, shown within Figure 5. Tag tables 76 and 86 are accessible by tag table managers 72 and 82, respectively. As will be more fully described below, tag table managers 72 and 82 perform several functions including the creation or deletion entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 may take form in software instructions executing on one or more processors of disk arrays 44 and 46, respectively.

[0029] Tag tables 76 and 86 are associated with mirrors M0 and M1, respectively. It is noted that additional tag tables like tag table 76 or 86 may be contained in memories 70 and 80, where each additional tag table is associated with a mirror of a mirrored data volume provided for access to application 52. However, for purposes of explanation, it will be presumed that memories 70 and 80 store only tag tables 76 and 86, respectively.

[0030] Each tag table 76 and 86 includes a plurality of entries. Each entry stores a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number. Alternatively, each tag table entry stores a tag and a physical block number or a range of disk block numbers beginning with a first disk block number. For purposes of explanation, it will be presumed that each entry of the tag tables 76 and 86 includes a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number unless otherwise noted. To illustrate, entry 2 of tag table 76 includes a tag equal to 2 and a logical block number equal to 25, and entry 3 of tag table 76 includes a tag equal to 70 and a range of 4 logical blocks beginning with logical memory block number 29.

[0031] Tag tables 76 and 86 track write IO transactions received by storage managers 62 and 64, respectively. Entries in tag tables 76 and 86 are created by tag managers 72 and 82, respectively, each time storage managers 62 and 64, respectively, receive IO transactions to

write data to mirrors M0 and M1, respectively. The contents (i.e., the tag and logical block number or range of logical block numbers) of the tag table entries are defined by the IO write transactions received by storage managers 62 and 64. To illustrate, presume storage manager 62 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M0. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 62 provides the tag and the logical block n or range of logical block numbers of the received IO transaction, to tag manager 72. Tag table manager 72, in response, creates a new entry in tag table 76. Tag table manager 72 then stores into the newly created entry of table 76 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 62 is written to the disk block or blocks in disk d0 allocated to, according to CMM0, the logical block n or range of logical blocks beginning with block n of mirror M0. It is noted that the foregoing process is also employed by tag table manager 82. More particularly, presume storage manager 64 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M1. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 64 provides the tag and the logical block n or range of logical block numbers to receiving the IO transaction, to tag manager 82. Tag table manager 82, in response, creates a new entry in tag table 86. Tag table manager 82 then stores into the newly created entry of table 86 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 64 is written to the disk block or blocks in disk d1 allocated to, according to CMM1, the logical block n or range of logical blocks beginning with block n of mirror M1.

[0032] Tag tables 76 and 86 can be used to determine whether mirrors M0 and M1 are in synchronization. If the data processing system 40 operates perfectly, e.g., there is no crash of host 42 or interruption of power to disk arrays 44 and 46, then all IO transactions to write data to mirrors M1 and M0 should reach disk arrays 46 and 44, respectively, and the contents of tag tables 76 and 86 should be identical. However, if due to, for example, crash of host 42 one of two separate but related IO transactions do not reach disk array 44 or 46, then the contents of tag tables 76 and 86 should differ and mirrors M0 and M1 will be out of synchronization. To illustrate, suppose host 42 generates first and second IO transactions for writing data D_{new} to logical block 30 in mirrors M0 and M1. Further, presume that each of these separate IO transactions includes a unique tag 51 generated by generator 54. Lastly, presume that disk array 44 receives the first IO transaction from host 42, but disk array 46 does not receive the second IO transaction. As a result, tag table manager 72 creates a new entry m in tag table 76 which includes tag 51 and block number 30 while no such corresponding tag entry is created in tag table 86. Tag table 76 indicates that logical block 30 of mirror M0 has been updated with data D_{new}. Since table 86 does not include a corresponding entry, it can be said that logical block 30 in mirror M1 has not been updated with data D_{new}, and mirrors M0 and M1 are out of synchronization.

[0033] In another embodiment, tokens can be generated by disk arrays 44 and 46 when IO transactions are received. To illustrate, presume storage managers 62 and 64 receive first and second IO transactions to write data to logical block n or a range of logical blocks beginning with block n of mirrors M0 and M1, respectively. Each of the first and second IO transactions include the same tag generated by tag generator 54 and which is unique to the first and second IO transactions. Storage manager 62 provides the tag and the logical block n or range of logical block numbers for the received first IO transaction, to tag manager 72. Likewise, storage manager 64 provides the tag and the logical block n or range of logical block numbers for the received second IO transaction, to tag manager 82. Tag table managers 72 and 82, in response, create new entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 then store into the newly created entries of tables 76 and 86 the tag and logical block n or range of logical block numbers beginning with logical block n. Thereafter each of tag managers 72 and 82 generate first and second tokens, respectively. The first and second tokens can be used to identify the newly created entries in tables 76 and 86, respectively. The tokens can be simple numbers to identify corresponding entries in the tag tables. In one embodiment, the first and second tokens can be created by, for example, concatenating the tags of the first and second IO transactions, respectively, with the same prefix. If and when storage managers 62 and 64 return status of the first and second IO transactions, respectively, to host 42, storage managers 62 and 64 may also return the first and second tokens associated with the newly generated tag entries in tables 76 and 86, respectively. The first and second tokens could be identical to each other, but they are not required to be identical. Host 42 receives status from both disk arrays 44 and 46, and host 42 declares the first and second IO transactions as complete. Host 42 will receive the first and second tokens. Presume now that host 42 generates third and fourth IO transactions to write data to block m of mirrors M0 and M1, respectively. Because host 42 received status reply for the first and second transactions, host 42 may add the first and second tokens to the third and fourth IO transactions before they are sent to disk arrays 44 and 46, respectively. Tag managers 72 and 82 are provided with the first and second tokens. Tag managers 72 and 82 delete entries in tables 76 and 86, respectively, that correspond to the first and second tokens, respectively, received from the host. In yet another alternative, host 42 can choose a certain time interval to send to disk arrays 44 and 46 a message consisting of several tokens, but only those tokens that are ready to be sent back to disk arrays 44 and 46. A token is ready to be sent back only after all related IO transactions are completed. The several tokens are subsequently provided to tag table managers 72 and 82, and tag managers 72 and 82 then delete entries corresponding to the several tokens.

[0034] When mirrors M0 and M1 are out of synchronization, table 76 and 86 can be used to bring mirrors M1 and M0 back into synchronization. Mirrors M0 and M1 will be out of synchronization when an entry is found within table 76 that is not matched in table 86, or when an entry is found in table 86 that is not matched in table 76. When this happens, data is copied from mirror M1 to mirror M0 or from mirror M0 to mirror M1 according to the table entry that lacks the matching entry. To illustrate, tables 76 and 86 are not identical because table 76 includes an entry m consisting of a tag number 51 and logical block 30, and this entry cannot be found within table 86. To bring these mirrors M0 and M1 back into synchronization, data is copied from logical block 30 of mirror M0 to logical block 30 of mirror M1. Thereafter, entry m is deleted from tag table 76. Once this copying process is complete, and assuming there are no other mismatching entries between tables 76 and 86, mirrors M1 and M0 are back in synchronization.

[0035] In addition to creating or deleting entries in tag tables 76 and 86, tag table managers 72 and 82 can operate to (1) identify instances where mirrors M0 and M1 are out of synchronization or (2) delete matching entries in tag tables 76 and 86. Figure 6 illustrates these operational aspects of tag table manager 72. The operational aspects shown in Figure 6 can be initiated at any time. For example, the process shown in Figure 6 can be initiated after failure of host 42.

[0036] After the process is started in step 90, tag table manager 72 selects an entry in tag table 76 as shown in step 94 and then sends a message to tag table manager 82 asking if there is a corresponding match within tag table 86. The message sent to tag table manager 82 identifies the tag and logical block number or range of logical block numbers of the tag selected in step 92. Tag table manager 82 accesses tag table 86 using tag and logical block number or range of logical block numbers of the entry selected in step 92 that is provided in the message sent by tag table manager 72, and determines if a match exists as shown in step 94. If tag table manager 82 finds an entry in table 86 that matches the entry selected in table 76 (i.e., an entry in table 86 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in step 92), then tag table manager deletes the matching entry in table 86 and sends a reply message to tag table manager 72 indicating that table 86 did include a match to the entry selected in step 92. Tag manager 72, in response to receiving the message from tag table manager 82 that a matching entry did exist, deletes the entry selected in step 92. The entry deletion by tag managers 72 and 82 is shown in step 96. However, if tag table 86 does not have a match to the entry selected in table 76, then a message to that effect is sent back to tag table manager 72 and the process proceeds to step 100. In step 100, data is copied from mirror M0 to mirror M1. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M0 that are identified by the entry selected in step 92, is copied to mirror M1 at the logical memory block or range of logical memory blocks identified by the entry selected in step 92. For example, assuming the selected entry in step 92 is entry m of table 76, the data contents of logical block 30 of mirror M0 is copied to logical block 30 of mirror M1 in step 100. After copying data from mirror M0 to mirror M1in step 100, tag table manager 72 deletes the entry selected in table 76 as shown in step 104. Thereafter, as shown in step 104, tag table manager 72 checks tag table 76 to see if any tag entries remain. If an

entry remains, then it is selected in step 92, and steps 94-102 are repeated. If no additional entries remain in table 76 at step 104, the process ends.

[0037] It is noted that tag table manager 82 operates under a process similar to that shown in Figure 6. In particular, tag table manager 82 selects an entry in tag table 86 and sends a message to tag table manager 72 asking if there is a corresponding match within tag table 76. The message sent to tag table manager 72 identifies the tag and logical block number or range of logical block numbers of the tag selected in table 86. Tag table manager 72 accesses tag table 76 using tag and logical block number or range of logical block numbers of the selected entry of table 86 provided in the message sent by tag table manager 82, and determines if a match exists in table 76. If tag table manager 72 finds an entry in table 76 that matches the entry selected in table 86 (i.e., an entry in table 76 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in table 86), then tag table manager 72 deletes the matching entry in table 76 and sends a reply message to tag table manager 82 indicating that table 76 did include a match to the entry selected in table 86. Tag manager 82, in response to receiving the message from tag table manager 72 that a matching entry did exist in table 76, deletes the selected entry in table 86. However, if tag table 76 does not have a match to the entry selected in table 86, then a message to that effect is sent to tag table manager 82 and data is copied from mirror M1 to mirror M0. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M1 that are identified by the entry selected in table 86 is copied to the logical memory block or range of logical memory blocks identified by the entry selected in table 86 but contained in mirror M0. After copying data from mirror M1 to mirror M0, tag table manager 82 deletes the selected entry in table 86. Thereafter, tag table manager 82 checks tag table 86 to see if any tag entries remain. If an entry remains, then it is selected, and the foregoing steps are repeated. If no additional entries remain in table 86, the process ends.

[0038] If host 42 crashes after sending a write IO transaction to disk array 44 but before sending the same IO transaction to disk array 46, that will leave M0 and M1 in inconsistent state. Before host 42 is restarted, synchronization authority will generate and send a message to disk arrays 44 and 46 instructing them to bring M0 and M1 into a consistent state as described by the foregoing sections.

[0039] While a resynchronization of mirrors M0 and M1 occur after, for example a crash of host 42, it may be necessary to suspend host 42 from further generation of IO transactions to write data to mirrors M0 and M1 until mirrors M0 and M1 are brought back into a consistent state.

[0040] As noted above, storage subsystems may take form in OSDs. In this embodiment, tag tables, such as tag tables 76 and 86, my have to be modified to include an object number and information indicating whether the corresponding write operation is overwriting existing data of the storage object or whether the corresponding write operation extends the file length.

[0041] It was noted above that in an alternative embodiment, the tag generator may be placed in one of the disk arrays 44 or 46. In this alternative embodiment, host 42 could forward the IO transaction from application 52 to disk array 44 (for example) that contains the tag generator with an instruction for disk array 44 to generate first and second write IO transactions that contain a unique tag. The first IO transaction would be provided to the storage manager 62 while the second IO transaction is transmitted to storage manager 64 in disk array 46.

[0042] Although the present invention has been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein. On the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the scope of the invention as defined by the appended claims.

WE CLAIM:

1. A method comprising:
a computer system generating first and second write transactions;
wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;
the computer system transmitting the first and second write transactions to first and second storage devices, respectively.

2. The method of claim 1 further comprising:
the computer system generating third and fourth write transactions;
wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;
the computer system transmitting the third and fourth write transactions to the first and second storage devices, respectively.

3. The method of claim 1 wherein:
the first write transaction comprises data D to be written to a logical block of a first storage object;
the second write transaction comprises data D to be written to a logical block of a second storage object.

4. The method of claim 3 further comprising:
the first storage device receiving the first write transaction;
the first storage device storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory;
the second storage device receiving the second write transaction;
the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

5. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

6. The method of claim 5 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an identity of the range of logical blocks of the first storage object where data D is to be written, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an identity of the range of logical blocks in the second storage object where data D is to be written, wherein the second tag table is stored in second memory.

7. The method of claim 4 further comprising comparing the contents of one entry in the first tag table with the contents of entries in the second tag table to determine whether the second tag table includes an entry that matches the one entry.

8. The method of claim 7 further comprising copying data from physical memory allocated to a logical block number identified in the one entry to physical memory allocated to the second storage object if the second table lacks an entry with contents matching the contents of the one entry.

9. The method of claim 7 further comprising deleting the one entry in the first table if the second table contains an entry with contents that match the contents of the one entry.

10. The method of claim 9 further comprising deleting the entry in the second table with contents that match the contents of the one entry.

11. The method of claim 1 further comprising:

the computer system generating a write transaction to write data to a logical block of a data volume;

the computer system incrementing a counter in response to generating the write transaction;

the computer system generating the first and second tags, wherein each of the first and second tags relate to the first and second write transactions, respectively, wherein the first and second tags are generated in response to generation of the write transaction, and wherein the first and second tags are generated as a function of an output of the incremented counter.

12. The method of claim 1 wherein the first and second storage devices comprise first and second object storage devices.

13. The method of claim 1 wherein:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

14. The method of claim 13 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an indication that data D is to be stored in the extension of the first storage object, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an indication that data D is to be stored in the extension of the second storage object, wherein the second tag table is stored in second memory.

15. A method comprising:

a computer system generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

the computer system transmitting the transaction to a storage device.

16. The method of claim 16 wherein the storage device stores a replication of the storage object.

17. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

generating first and second write transactions;

wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;

transmitting the first and second write transactions directly or indirectly to first and second storage devices, respectively.

18. The computer readable medium of claim 17 wherein the method further comprises:

generating third and fourth write transactions;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;

transmitting the third and fourth write transactions directly or indirectly to the first and second storage devices, respectively.

19. The computer readable medium of claim 17 wherein:

the first write transaction comprises data D to be written to a logical block of a first storage object;

the second write transaction comprises data D to be written to a logical block of a second storage object.

20. The computer readable medium of claim 17 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

21. The computer readable medium of claim 17 wherein the first tag is identical to the second tag.

22. The computer readable medium of claim 17 wherein the method further comprises:

generating a write transaction to write data to a logical block of a data volume;

incrementing a counter in response to generating the write transaction;

generating the first and second tags, wherein each of the first and second tags relate to the first and second write transactions, respectively, wherein the first and second tags are generated in response to generation of the write transaction, and wherein the first and second tags are generated as a function of an output of the incremented counter,.

23. The computer readable medium of claim 17 the first and second storage devices comprise first and second object storage devices.

24. The computer readable medium of claim 17:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

25. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

transmitting the transaction to a storage device.

26. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

in response to receiving a first transaction comprising a first tag, storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory, wherein the first tag corresponds to a second tag of a second write transaction;

the second storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIV E VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

ABSTRACT OF THE DISCLOSURE

[0043] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

DECLARATION FOR PATENT APPLICATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below adjacent to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of subject matter (process, machine, manufacture, or composition of matter, or an improvement thereof) which is claimed and for which a patent is sought by way of the application entitled

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

which (check) ☒ is attached hereto.
□ and is amended by the Preliminary Amendment attached hereto.
□ was filed on ___ as Application Serial No. _____
□ and was amended on (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
Number	Country	Day/Month/Year Filed	Yes	No
			□	□

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

Provisional Application Number	Filing Date

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) or PCT international application(s) designating the United States of America listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal

Regulations, § 1.56, which became available between the filing date of the prior application(s) and the national or PCT international filing date of this application:

Application Serial No.	Filing Date	Status (patented, pending, abandoned)

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith:

Joseph FitzGerald (33,881); John Brigden (40,530); Julie Stephenson (41,330); and

Marc R. Ascolese (42,268); Brenna A. Brock (48,309); Sam G. Campbell (42,381); Justin M. Dillon (42,486); D'Ann Rifai (47,026); Eric A. Stephenson (38,321).

Please address all correspondence and telephone calls to:

Eric A. Stephenson
CAMPBELL STEPHENSON ASCOLESE LLP
4807 Spicewood Springs Road
Building 4, Suite 201
Austin, Texas 78759
Telephone: 512-439-5093
Facsimile: 512-439-5099

I declare that all statements made herein of my own knowledge are true, all statements made herein on information and belief are believed to be true, and all statements made herein are made with the knowledge that whoever, in any matter within the jurisdiction of the Patent and Trademark Office, knowingly and willfully falsifies, conceals, or covers up by any trick, scheme, or device a material fact, or makes any false, fictitious or fraudulent statements or representations, or makes or uses any false writing or document knowing the same to contain any false, fictitious or fraudulent statement or entry, shall be subject to the penalties including fine or imprisonment or both as set forth under 18 U.S.C. 1001, and that violations of this paragraph may jeopardize the validity of the application or this document, or the validity or enforceability of any patent, trademark registration, or certificate resulting therefrom.

Full name of sole (or first joint inventor): Ronald S. Karr

Inventor's Signature: ______________________ Date: ______________

Residence: Palo Alto, California 94301

Post Office Address: 333 Ramona Street
Palo Alto, California 94301

Citizenship: U.S.A.

Full name of second inventor: Ramana Jonnala

Inventor's Signature: ______________________ Date: ______________

Residence: Sunnyvale, California 94086

Post Office Address: 825 E. Evelyn Avenue, #326
Sunnyvale, California 94086

Citizenship: U.S.A.

Full name of third inventor: Narasimha R. Valiveti

Inventor's Signature: ______________________ Date: ______________

Residence: Sunnyvale, California 94085

Post Office Address: 395 Ano Nuevo Avenue, Apt #413
Sunnyvale, California 94085

Citizenship: India

Full name of fourth inventor: Dhanesh Joshi

Inventor's Signature: ______________________ Date: ______________

Residence: Santa Clara, California 95051

Post Office Address: 46, Cabot Avenue
Santa Clara, California 95051

Citizenship: India

EXHIBIT Q

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:57 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Final Draft And Declaration
Attachments: VRT0120US-Patent Application.DOC; VRT0120US-Declaration.doc

---

From: Eric Stephenson
Sent: Thursday, February 26, 2004 3:37 PM
To: 'Dhanesh Joshi'; Ron Karr; Narasimha Valiveti; Ramana Jonnala
Cc: julie.stephenson@veritas.com
Subject: RE: VRTS 0655 (VRT0120US) Final Draft And Declaration

Thanks Dhanesh,

I've revised the specification in accordance with your comments below. The revised application is attached.

If all is in order, I ask all the inventors to sign the attached declaration.

Thanks,

Eric

> -----Original Message-----
> **From:** Dhanesh Joshi [mailto:Dhanesh@veritas.com]
> **Sent:** Thursday, February 26, 2004 2:22 PM
> **To:** Eric Stephenson; Ron Karr; Narasimha Valiveti; Ramana Jonnala
> **Subject:** RE: VRTS 0655 (VRT0120US) Final Draft And Declaration
>
> Hi Eric,
> sorry for not being very clear in my previous mail. A small correction -
>
> > Thereafter each of tag managers 72 and 82 generate first and second tokens by, for example, concatenating the tags
> > of the first and second IO transactions
>
> Each of the tag managers 72 and 82 generate the first and second token respectively. These token basically used to identify the tag entry in the tag table. These tokens can even be some simple numbers to identify the tag entry.
>
> > The first and second tokens should be identical to each other.
> This is not correct. As 72 and 82 may generate the tokens by any algorithm that they like, they may or may not be the same. This logic does not require them to be the same. No one, except the tag manager that generated a particular token, tries to interpret the token. i.e. 72 only understands/interprets the token that it generated.
>
> > At this point host 42 knows that mirrors M0 and M1 are consistent with each other, at least with respect to the first

> and second IO transactions, since host 42 receives the matching first and second tokens.

Host 42 interprets the "result" of the transaction but it does not interpret the token. 42 does not care if they are matching or not. If the returned status from both, 72 and 82 is SUCCESS, then it knows that M0 and M1 are in sync.

> Presume now that host 42 generates third and fourth IO transactions to write data to block m of mirrors M0 and M1,
> respectively. Because host 42 has received the matching first and second tokens, host 42 could add the first and
> second tokens to the third and fourth IO transactions before they are sent to disk arrays 44 and 46, respectively.

A token is considered to be ready for "piggyback" only after all related IO transaction are completed. For example,
consider that host 42 starts 3rd and 4th transaction before both 1st and 2nd are completed. Lets consider at this time that 1st is completed but second is not. So 42 has status and token of the 1st transaction. In this case it will not piggyback 1st token with the 3rd transaction.

> Tag managers 72 and 82 are provided with the first and second tokens, respectively, and remove the prefix.
> Tag managers 72 and 82 then compare the tags that remain after removing the prefix to the tags of entries in tables 76
> and 86, respectively. When matches are found in the tag tables, the corresponding entries are removed.

I did not understand what you meant by "remove prefix".

Tag managers 72 and 82 receives the piggybacked tokens. When they see such token they "know" that IO transaction
associated with that token has reached to the all "other" mirror as well and they can safely remove the tag entry in the table. 72 and 82 uses these tokens to "locate" the tag entry in the tag table.

thanks
-- dhanesh

EXHIBIT R

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:57 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Final Draft And Declaration

---

From: Dhanesh Joshi [mailto:Dhanesh@veritas.com]
Sent: Thursday, February 26, 2004 4:35 PM
To: Eric Stephenson
Subject: RE: VRTS 0655 (VRT0120US) Final Draft And Declaration

Hi Eric
claim 8 refers disk blocks with "physical memory".

> The method of claim 7 further comprising copying data from physical memory allocated to a logical block number
> identified in the one entry to physical memory allocated to the second storage object if the second table lacks an
> entry with contents matching the contents of the one entry.

This should something like -
The method of claim 7 further comprising copying data, associated with the logical block number identified by the one entry, from the first storage object to the the logical block in the second storage object if the second table lacks an entry with contents matching the contents of the one entry.

One more thing is that following lines -

"In one embodiment, the first and second tokens can be created by, for example, concatenating the tags of the
first and second IO transactions, respectively, with the same prefix."

should be removed from the section [0033]

Claim 16 -
What is claim 16 ? I could not understand that claim
I noted that claim 16 refers to itlesf.

-- dhanesh

> -----Original Message-----
> **From:** Eric Stephenson [mailto:estephenson@csapatent.com]
> **Sent:** Thursday, February 26, 2004 1:37 PM
> **To:** Dhanesh Joshi; Ron Karr; Narasimha Valiveti; Ramana Jonnala
> **Cc:** Julie Stephenson
> **Subject:** RE: VRTS 0655 (VRT0120US) Final Draft And Declaration
>
> Thanks Dhanesh,

I've revised the specification in accordance with your comments below. The revised application is attached.

If all is in order, I ask all the inventors to sign the attached declaration.

Thanks,

Eric

-----Original Message-----
From: Dhanesh Joshi [mailto:Dhanesh@veritas.com]
Sent: Thursday, February 26, 2004 2:22 PM
To: Eric Stephenson; Ron Karr; Narasimha Valiveti; Ramana Jonnala
Subject: RE: VRTS 0655 (VRT0120US) Final Draft And Declaration

Hi Eric,
sorry for not being very clear in my previous mail. A small correction -

> Thereafter each of tag managers 72 and 82 generate first and second tokens by, for example, concatenating the tags
> of the first and second IO transactions

Each of the tag managers 72 and 82 generate the first and second token respectively. These token basically used to identify the tag entry in the tag table. These tokens can even be some simple numbers to identify the tag entry.

> The first and second tokens should be identical to each other.
This is not correct. As 72 and 82 may generate the tokens by any algorithm that they like, they may or may not be the same. This logic does not require them to be the same. No one, except the tag manager that generated a particular token, tries to interpret the token. i.e. 72 only understands/interprets the token that it generated.

> At this point host 42 knows that mirrors M0 and M1 are consistent with each other, at least with respect to the first
> and second IO transactions, since host 42 receives the matching first and second tokens.

Host 42 interprets the "result" of the transaction but it does not interpret the token. 42 does not care if they are matching or not. If the returned status from both, 72 and 82 is SUCCESS, then it knows that M0 and M1 are in sync.

> Presume now that host 42 generates third and fourth IO transactions to write data to block m of mirrors M0 and M1,
> respectively. Because host 42 has received the matching first and second tokens, host 42 could add the first and
> second tokens to the third and fourth IO transactions before they are sent to disk arrays 44 and 46, respectively.

A token is considered to be ready for "piggyback" only after all related IO transaction are completed. For example,
consider that host 42 starts 3rd and 4th transaction before both 1st and 2nd are completed. Lets consider at this time that 1st is completed but second is not. So 42 has status and token of the 1st transaction. In this case it will not piggyback 1st token with the 3rd transaction.

> Tag managers 72 and 82 are provided with the first and second tokens, respectively, and

remove the prefix.
> Tag managers 72 and 82 then compare the tags that remain after removing the prefix to the tags of entries in tables 76
> and 86, respectively. When matches are found in the tag tables, the corresponding entries are removed.

I did not understand what you meant by "remove prefix".

Tag managers 72 and 82 receives the piggybacked tokens. When they see such token they "know" that IO transaction
associated with that token has reached to the all "other" mirror as well and they can safely remove the tag entry in the table. 72 and 82 uses these tokens to "locate" the tag entry in the tag table.

thanks
-- dhanesh

EXHIBIT S

EXPRESS MAIL LABEL NO.:

(EV 304738055 US)

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

BACKGROUND OF THE INVENTION

[0001] Large scale data processing systems typically include several data storage subsystems (e.g., disk arrays) each containing many physical storage devices (e.g., hard disks) for storing critical data. Data processing systems often employ storage management systems to aggregate these physical storage devices to create highly reliable data storage. There are many types of highly reliable storage. Mirrored volume storage is an example. Mirrored volume storage replicates data over two or more mirrors of equal size. A logical memory block n of a mirrored volume maps to the same logical memory block n of each mirror. In turn, each logical memory block n of the mirrors map directly or indirectly to one or more disk blocks of one or more physical devices. Mirrored volumes provide data redundancy. If an application is unable to access data of one, a duplicate of the data sought should be available in an alternate mirror.

[0002] Figure 1 illustrates relevant components of an exemplary data processing system 10 that employs a two-way mirrored volume. While the present invention will be described with reference to a two-way mirrored volume, the present invention should not be limited thereto. The present invention may find use with other types of redundant storage including, for example, a three-way mirrored storage volume. Data processing system 10 includes a host (e.g., server computer system) 12 coupled to data storage subsystems 14 and 16 via storage interconnect 20. For purposes of explanation, storage interconnect 20 will take form in a storage area network (SAN) it being understood that the term storage interconnect should

not be limited thereto. SAN 20 may include devices (e.g., switches, routers, hubs, etc.) that cooperate to transmit input/output (IO) transactions between host 12 and storage subsystems 14 and 16.

[0003] Each of the data storage subsystems 14 and 16 includes several physical storage devices. For purposes of explanation, data storage subsystems 14 and 16 are assumed to include several hard disks. The term physical storage device should not be limited to hard disks. Data storage subsystems 14 and 16 may take different forms. For example, data storage subsystem 14 may consist of "Just a Bunch of Disks" (JBOD) connected to an array controller card. Data storage subsystem 16 may consist of a block server appliance. For purposes of explanation, each of the data storage subsystems 14 and 16 will take form in an intelligent disk array, it being understood that the term data storage subsystem should not be limited thereto.

[0004] As noted, each of the disk arrays 14 and 16 includes several hard disks. The hard disk is the most popular permanent storage device currently used. A hard disk's total storage capacity is divided into many small chunks called physical memory blocks or disk blocks. For example, a 10 GB hard disk contains 20 million disk blocks, with each block able to hold 512 bytes of data. Any random disk block can be written to or read from in about the same time, without first having to read or write other disk blocks. Once written, a disk block continues to hold data even after the hard disk is powered down. While hard disks in general are reliable, they are subject to occasional failure. Data systems employ data redundancy schemes such as mirrored volumes to protect against occasional failure of a hard disk.

[0005] Host 12 includes an application 22 executing on one or more processors. Application 22 generates IO transactions to access critical data in response to receiving requests from client computer systems (not shown) coupled to host 12. In addition to application 22, host 12 includes a storage manager 24 executing on one or more processors. Volume Manager^{TM} provided by VERITAS Software Corporation of Mountain View, California, is an exemplary storage manager. Although many of the examples described herein will emphasize virtualization architecture and terminology associated with the VERITAS Volume Manager™, the software and techniques described herein can be used with a variety of different storage managers and architectures.

[0006] Storage managers can perform several functions. More particularly, storage managers can create storage objects (also known as virtualized disks) by aggregating hard disks such as those of disk arrays 14 and 16, underlying storage objects, or both. A storage object is an abstraction. Figure 2 shows a visual representation of exemplary storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ created for use in data processing system 10. Each of the storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ in Figure 2 consists of an array of n_{max} logical memory blocks that store or are configured to store data. While it is said that a logical memory block stores or is configured to store data, in reality the data is stored in one or more disk blocks of hard disks allocated directly or indirectly to the logical memory block.

[0007] Storage objects aggregated from hard disks can themselves be aggregated to form storage objects called logical data volumes. Logical data volumes are typically presented for direct or indirect use by an application such as application 22 executing on host 12. Thus, application 22 generates IO transactions to read data from or write data to one or more logical memory blocks of a data volume not knowing that the data volume is an aggregation of underlying storage objects, which in turn are aggregations of hard disks. Properties of storage objects depend on how the underlying storage objects or hard disks are aggregated. In other words, the method of aggregation determines the storage object type. In theory, there are a large number of possible methods of aggregation. The more common forms of aggregation include concatenated storage, striped storage, RAID storage, or mirrored storage. A more thorough discussion of how storage objects or hard disks can be aggregated can be found within Dilip M. Ranade [2002], "Shared Data Clusters" Wiley Publishing, Inc., which is incorporated herein by reference in its entirety.

[0008] Mirrored volumes provide highly reliable access to critical data. $V_{Example}$ of Figure 2 is an exemplary two-way mirrored volume. $V_{Example}$ was created by aggregating underlying storage objects (hereinafter mirrors) $M0_{Example}$ and $M1_{Example}$. Mirrors $M0_{Example}$ and $M1_{Example}$ were created by concatenating disk blocks from hard disks $d0_{Example}$ and $d1_{Example}$ (not shown) in disk arrays 14 and 16, respectively.

[0009] Storage managers can create storage object descriptions that describe the relationship between storage objects and their underlying storage objects or hard disks. These storage object descriptions typically include configuration maps. It is noted that

storage object descriptions may include other information such as information indicating that a storage object is a snapshot copy of another storage object.

[0010] A configuration map maps a logical memory block of a corresponding storage object to one or more logical memory blocks of one or more underlying storage objects or to one or more disk blocks of one or more hard disks. To illustrate, configuration maps $CMV_{Example}$, $CMM0_{Example}$, and $CMM1_{Example}$ are created for mirrored volume $V_{Example}$ and underlying mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Configuration map $CMV_{Example}$ maps each logical memory block n of $V_{Example}$ to logical memory blocks n of mirrors $M0_{Example}$ and $M1_{Example}$. Configuration map $CMM0_{Example}$ maps each logical memory block n of mirror $M0_{Example}$ to a disk block x in hard disk $d0_{Example}$, while configuration map $CMM1_{Example}$ maps each logical memory block n of mirror $M1_{Example}$ to a disk block y in hard disk $d1_{Example}$. Configuration map $CMV_{Example}$ can be provided for use by storage manager 24, while configuration maps $CMM0_{Example}$ and $CMM1_{Example}$ can be provided for use by storage managers 34 and 36 executing on one or more processors in disk arrays 14 and 16, respectively.

[0011] Storage managers use configuration maps to translate IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks. To illustrate, presume an IO transaction is generated by application 22 to write data D to logical memory block 3 of data volume $V_{Example}$. This IO transaction is received directly or indirectly by storage manager 24. In turn, storage manager 24 accesses configuration map $CMV_{Example}$ and learns that logical memory block 3 is mapped to logical block 3 in both mirrors $M0_{Example}$ and $M1_{Example}$. It is noted storage manager may not receive the exact IO transaction generated by application 22. However, the transaction storage manager 24 receives will indicate that data D is to be written to block 3 of mirrored volume $V_{Example}$.

[0012] Storage manager 24 then generates first and second IO transactions to write data D to logical blocks 3 and mirrors $M0_{Example}$ and $M1_{Example}$, respectively. The IO transactions generated by storage manager 24 are transmitted to disk arrays 14 and 16, respectively, via SAN 20. Storage managers 34 and 36 of disk arrays 14 and 16, respectively receive directly or indirectly, the IO transactions sent by storage manager 24. It is noted that the IO

transactions received by storage managers 34 and 36 may not be the exact IO transactions generated and sent by storage manager 24. Nonetheless, storage managers 34 and 36 will each receive an IO transaction to write data D to logical block 3 in mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Storage manager 34, in response to receiving the IO transaction, accesses configuration map $CMM0_{Example}$ to learn that block 3 of mirror $M0_{Example}$ is mapped to, for example, disk block 200 within hard disk d0. In response, a transaction is generated to write data D to disk block 200 within disk d0. Storage manager 36 accesses configuration map $CMM1_{Example}$ to learn that logical block 3 of mirror $M1_{Example}$ is mapped to, for example, disk block 300 within hard disk d1. In response, an IO transaction is generated for writing data D to disk block 300 within hard disk d1.

[0013] As noted above, while hard disks are reliable, hard disks are subject to failure. Data may be inaccessible within a failed hard disk. For this reason and others, administrators create mirrored data volumes. Unfortunately, data within mirrors of a mirrored volume may get into an inconsistent state if, for example, there is a server crash, storage power failure, or other problem which prevents data from being properly written to a hard disk. Consider the exemplary mirrored volume $V_{Example}$. Presume storage manager 24 generates first and second IO transactions to write data D to logical blocks 3 in mirrors $M0_{Example}$ and $M1_{Example}$ in response to the IO transaction generated by application 22. Further, presume host 12 may fail after the first IO transaction is transmitted to disk array 14, but before the second IO transaction is transmitted to disk array 16. As a result, data D is written to disk block 200 within disk d0 of disk array 14, but data D is not written to disk block 300 within hard disk d1. When host 12 is restarted and exemplary volume $V_{Example}$ is made available again to application 22, mirrors $MO_{Example}$ and $M1_{Example}$ are said to be out of sync. In other words, mirrors $M0_{Example}$ and $M1_{Example}$ are no longer identical since at least block 3 in each contain different data. An IO transaction to read from logical memory block 3 of mirrored volume $V_{Example}$ could return either old or new data depending on whether the data is read from disk block 200 of hard disk d0 or disk block 300 of hard disk d1. Mirrors $MO_{Example}$ and $M1_{Example}$ should be resychronized before either is accessed again.

[0014] A brute force method to resynchronize mirrors $M0_{Example}$ and $M1_{Example}$ is simply to presume that one mirror (e.g., $M0_{Example}$) contains correct data and copy the contents of the one mirror to the other (e.g., $M1_{Example}$). It can take hours to resynchronize using this method.

A smarter resynchronization technique is possible, but it requires some preparation. This alternate technique involves using what is called a dirty region map. Figure 2 illustrates a visual representation of an exemplary dirty region map 40 consisting of n_{max} entries corresponding to the n_{max} logical memory blocks within mirrors $M0_{Example}$ and $M1_{Example}$. Each entry of the dirty region map 40 indicates whether the corresponding blocks in mirrors are considered synchronized. For example, if entry n is set to logical 1, then blocks n in the mirrors are considered out of synchronization, and if n is set to logical 0, blocks n in the mirrors are considered in synchronization. Entry n in dirty region map 40 is set to logical 1 when the application 22 generates a transaction for writing data to logical block n of volume $V_{Example}$. Entry n is maintained in the logical 1 state until acknowledgement is received from both disk arrays 14 and 16 that data D has been written successfully to the disk blocks allocated to logical memory block n. However, if, using the example above, the first IO transaction to write data D to block 3 of mirror $M0_{Example}$ succeeds while the second IO transaction to write to block 3 of $M1_{Example}$ fails, then entry 3 and dirty region map 40 will be maintained as a logical 1 indicating that logical blocks 3 in the mirrors are out of synchronization. Mirrors $M0_{Example}$ and $M1_{Example}$ can be resynchronized by copying data from $M0_{Example}$ to mirror $M1_{Example}$, but for only logical memory blocks corresponding to dirty region map entries set to logical 1.

SUMMARY OF THE INVENTION

[0015] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

Fig. 1 illustrates a block diagram of a data system in which a logical data volume may be created;

Fig. 2 illustrates a visual representation of the storage objects created in the data system of Fig. 1;

Fig. 3, illustrates a block diagram of a data system capable of implementing one embodiment of the present invention;

Figs. 4 shows visual representations of a mirrored data volume employed in the data system of Fig. 3;

Fig. 5 is a visual representation of exemplary tag tables employed in the disk arrays of Fig. 3;

Fig.6 is a flow chart illustrating relevant aspects of managing the tag tables shown in Fig. 5.

[0017] The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION

[0018] Figure 3 illustrates in block diagram form, relevant components of an exemplary data processing system 40 capable of employing one embodiment of the present invention. Data processing system 40 includes a host (e.g., server computer system) 42 coupled to data storage subsystems 44 an 46 via storage interconnect 50. The present invention should not be limited to use in a data processing system consisting of only two storage subsystems. The present invention may be employed in a data processing system consisting of more than two storage subsystems.

[0019] Storage interconnect 50 takes form in a SAN consisting of several devices (e.g., switches, routers, etc.) that operate to transmit data between devices 42-46 including IO transactions to read or write data, it being understood that the term storage interconnect should not be limited thereto. SAN 50 can operate according to any one of a number of different protocols including Fibre Channel, Ethernet, etc. Each of the devices 42-46 includes one or more processors capable of processing data according to instructions of a software component such as a storage manager. As such, each of the devices 42-46 can be considered a computer system.

[0020] Data storage subsystems 44 and 46 may take different forms. For example, data storage subsystems 44 and 46 may take form in object storage devices (OSDs). Unless otherwise noted, each of the data storage subsystems 44 and 46 is implemented as a disk array, it being understood that the term data storage subsystem should not be limited thereto. Each of the disk arrays 44 and 46 includes several hard disks. Moreover, each of the disk arrays 44 and 46 may include a memory for storing a tag table that will be more fully described below.

[0021] Host 42 includes an application 52 executing on one or more processors. Application 52 may take one of many different forms. For example, application 52 may take form in a database management system (DBMS), customer relationship management software, etc. Application 52 generates IO transactions to read data from or write data to a data volume in response to receiving requests from client computer systems (not shown) coupled to host 42. Host 42 also includes a tag generator 54 executing on one or more

processors. It is noted that tag generator, in an alternative embodiment, could be placed in a different device of system 40. For example, tag generator 54 could be a part of one of the disk arrays 44 or 46. For purposes of explanation, it will be presumed that the tag generator is placed in host 42 as shown in Figure 4, unless otherwise noted. Tag generator 54 will be more fully described below.

[0022] Devices 42-46 include a storage manager 60-64, respectively, executing on one or more processors. Each of the storage managers 60-64 is capable of performing many functions. More particularly, each of the storage managers 60-64 is capable of translating IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks of disk arrays 44 and 46. Storage managers 60-64 translate IO transactions according to configuration maps provided thereto. For purposes of illustration, it will be presumed that application 52 is provided with access to a mirrored data volume V having two mirrors, storage objects M0 and M1. It is noted that the present invention can be employed with respect to a data volume consisting of more than two mirrors.

[0023] A visual representation of volume V and mirrors M0 and M1 is found within Figure 4. As shown in Figure 4, each of V, M0, and M1 consist of an array of n_{max} logical memory blocks. It is noted that mirrors M0 and M1 could be formed from underlying storage objects (e.g., logical units or LUNs), or that mirrors M0 and M1 could be created as striped or RAID storage over hard disks or LUNs. It will be presumed, however, that M0 is formed by aggregating disk blocks contained within disk d0 (not shown) of disk array 44, and that mirror M1 is formed by aggregating disk blocks contained within disk d1 of disk array 46.

[0024] Storage managers 60-64 are provided with storage object descriptions for storage objects V, M0, and M1, respectively. Generally, the storage object descriptions define the relationships between storage objects and their underlying storage objects or hard disks. Moreover, the storage object descriptions may include other information. For example, the storage object descriptions for M0 and M1 indicate that M0 and M1 are mirrors or each other. As such, storage manager 62 knows that a mirror of storage object M0 is contained within storage subsystem 46, and storage manager 64 knows that a mirror of storage object M1 is contained within storage subsystem 44.

[0025] Each of the storage object descriptions may also include a configuration map. A configuration map CMV is provided to storage manager 60 which maps each logical block n of volume V to corresponding blocks n in mirrors M0 and M1. Storage manager 62 is provided with a configuration map CMM0 which maps each logical block n of mirror M0 to a disk block x within hard disk d0. Lastly, storage manager 64 is provided with a configuration map CMM1 which maps each logical block n of mirror M1 to a disk block y within hard disk d1. Storage manager 60 is capable of translating IO transactions received from application 52 using configuration map CMV. Thus, when application 52 generates a transaction to write a data to block n of volume V, storage manager in response generates first and second transactions for writing a data D to blocks n in mirrors M0 and M1, respectively. Storage manager 62, in response to receiving an IO transaction to write data to block n of mirror M0, uses its configuration map CMM0 to identify the corresponding disk block x within disk d0 to which data D is to be written. Likewise, storage manager 64, in response to receiving an IO transaction to write data D to block n of mirror M1, uses its configuration map CMM1 to identify the corresponding disk block y within disk d1 to which data D is to be written.

[0026] IO transactions are received by storage manager 60. Tag generator 54, in response to storage manager 60 receiving an IO transaction to write data, generates a tag unique to the IO transaction to write data. This tag is provided to storage manager 60. In one embodiment, the tag may be the count value of a counter. In this embodiment, the count value is incremented by one each time storage manager 60 receives an IO transaction to write data.

[0027] Storage manager 60 accesses its configuration map CMV in response to receiving an IO transaction to write data to logical memory block n of volume V or a range of logical memory blocks beginning with block n. For purposed of explanation, it will be presumed that storage manager 60 receives IO transactions to write data to one logical block n of volume V unless otherwise noted. From configuration map CMV, storage manager 60 learns that logical block n of volume V is mapped to a logical blocks n in mirrors M1 and M0. Accordingly, storage manager 60 generates first and second IO transactions to write data D to block n of mirrors M0 and M1. Each of these first and second IO transactions also includes the unique tag generated by tag generator 54. The first and second IO transactions, including

the unique tag, are transmitted by host 42 to disk arrays 44 and 46, respectively, via SAN 50. It is noted that the protocol employed with SAN 50 may have to be modified to accommodate IO transactions that include a unique tag.

[0028] Storage manager 62 is in data communication with nonvolatile memory 70. Likewise, storage manager 64 is in data communication with nonvolatile memory 80. Nonvolatile memories 70 and 80 are configured to store tag tables, such as tag tables 76 and 86, respectively, shown within Figure 5. Tag tables 76 and 86 are accessible by tag table managers 72 and 82, respectively. As will be more fully described below, tag table managers 72 and 82 perform several functions including the creation or deletion entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 may take form in software instructions executing on one or more processors of disk arrays 44 and 46, respectively.

[0029] Tag tables 76 and 86 are associated with mirrors M0 and M1, respectively. It is noted that additional tag tables like tag table 76 or 86 may be contained in memories 70 and 80, where each additional tag table is associated with a mirror of a mirrored data volume provided for access to application 52. However, for purposes of explanation, it will be presumed that memories 70 and 80 store only tag tables 76 and 86, respectively.

[0030] Each tag table 76 and 86 includes a plurality of entries. Each entry stores a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number. Alternatively, each tag table entry stores a tag and a physical block number or a range of disk block numbers beginning with a first disk block number. For purposes of explanation, it will be presumed that each entry of the tag tables 76 and 86 includes a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number unless otherwise noted. To illustrate, entry 2 of tag table 76 includes a tag equal to 2 and a logical block number equal to 25, and entry 3 of tag table 76 includes a tag equal to 70 and a range of 4 logical blocks beginning with logical memory block number 29.

[0031] Tag tables 76 and 86 track write IO transactions received by storage managers 62 and 64, respectively. Entries in tag tables 76 and 86 are created by tag managers 72 and 82, respectively, each time storage managers 62 and 64, respectively, receive IO transactions to

write data to mirrors M0 and M1, respectively. The contents (i.e., the tag and logical block number or range of logical block numbers) of the tag table entries are defined by the IO write transactions received by storage managers 62 and 64. To illustrate, presume storage manager 62 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M0. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 62 provides the tag and the logical block n or range of logical block numbers of the received IO transaction, to tag manager 72. Tag table manager 72, in response, creates a new entry in tag table 76. Tag table manager 72 then stores into the newly created entry of table 76 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 62 is written to the disk block or blocks in disk d0 allocated to, according to CMM0, the logical block n or range of logical blocks beginning with block n of mirror M0. It is noted that the foregoing process is also employed by tag table manager 82. More particularly, presume storage manager 64 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M1. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 64 provides the tag and the logical block n or range of logical block numbers to receiving the IO transaction, to tag manager 82. Tag table manager 82, in response, creates a new entry in tag table 86. Tag table manager 82 then stores into the newly created entry of table 86 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 64 is written to the disk block or blocks in disk d1 allocated to, according to CMM1, the logical block n or range of logical blocks beginning with block n of mirror M1. It is noted that in another embodiment, host 42 includes multiple, independent applications each of which is capable of generating a transaction for writing data. In this embodiment, each application may have its own tag generator that generates tags unique to that tag generator. Further in this embodiment, the tag manager of each disk array is capable of distinguishing between tags generated by different tag generators.

[0032] Tag tables 76 and 86 can be used to determine whether mirrors M0 and M1 are in synchronization. If the data processing system 40 operates perfectly, e.g., there is no crash of host 42 or interruption of power to disk arrays 44 and 46, then all IO transactions to write data to mirrors M1 and M0 should reach disk arrays 46 and 44, respectively, and the contents of tag tables 76 and 86 should be identical. However, if due to, for example, crash of host 42 one of two separate but related IO transactions do not reach disk array 44 or 46, then the contents of tag tables 76 and 86 should differ and mirrors M0 and M1 will be out of synchronization. To illustrate, suppose host 42 generates first and second IO transactions for writing data D_{new} to logical block 30 in mirrors M0 and M1. Further, presume that each of these separate IO transactions includes a unique tag 51 generated by generator 54. Lastly, presume that disk array 44 receives the first IO transaction from host 42, but disk array 46 does not receive the second IO transaction. As a result, tag table manager 72 creates a new entry m in tag table 76 which includes tag 51 and block number 30 while no such corresponding tag entry is created in tag table 86. Tag table 76 indicates that logical block 30 of mirror M0 has been updated with data D_{new}. Since table 86 does not include a corresponding entry, it can be said that logical block 30 in mirror M1 has not been updated with data D_{new}, and mirrors M0 and M1 are out of synchronization.

[0033] In another embodiment, tokens can be generated by disk arrays 44 and 46 when IO transactions are received. To illustrate, presume storage managers 62 and 64 receive first and second IO transactions to write data to logical block n or a range of logical blocks beginning with block n of mirrors M0 and M1, respectively. Each of the first and second IO transactions include the same tag generated by tag generator 54 and which is unique to the first and second IO transactions. Storage manager 62 provides the tag and the logical block n or range of logical block numbers for the received first IO transaction, to tag manager 72. Likewise, storage manager 64 provides the tag and the logical block n or range of logical block numbers for the received second IO transaction, to tag manager 82. Tag table managers 72 and 82, in response, create new entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 then store into the newly created entries of tables 76 and 86 the tag and logical block n or range of logical block numbers beginning with logical block n. Thereafter each of tag managers 72 and 82 generate first and second tokens, respectively.

The first and second tokens can be used to identify the newly created entries in tables 76 and 86, respectively. The tokens can be simple numbers to identify corresponding entries in the tag tables. If and when storage managers 62 and 64 return status of the first and second IO transactions, respectively, to host 42, storage managers 62 and 64 may also return the first and second tokens associated with the newly generated tag entries in tables 76 and 86, respectively. The first and second tokens could be identical to each other, but they are not required to be identical. Host 42 receives status from both disk arrays 44 and 46, and host 42 declares the first and second IO transactions as complete. Host 42 will receive the first and second tokens. Presume now that host 42 generates third and fourth IO transactions to write data to block m of mirrors M0 and M1, respectively. Because host 42 received status reply for the first and second transactions, host 42 may add the first and second tokens to the third and fourth IO transactions before they are sent to disk arrays 44 and 46, respectively. Tag managers 72 and 82 are provided with the first and second tokens. Tag managers 72 and 82 delete entries in tables 76 and 86, respectively, that correspond to the first and second tokens, respectively, received from the host. In yet another alternative, host 42 can choose a certain time interval to send to disk arrays 44 and 46 a message consisting of several tokens, but only those tokens that are ready to be sent back to disk arrays 44 and 46. A token is ready to be sent back only after all related IO transactions are completed. The several tokens are subsequently provided to tag table managers 72 and 82, and tag managers 72 and 82 then delete entries corresponding to the several tokens.

[0034] When mirrors M0 and M1 are out of synchronization, table 76 and 86 can be used to bring mirrors M1 and M0 back into synchronization. Mirrors M0 and M1 will be out of synchronization when an entry is found within table 76 that is not matched in table 86, or when an entry is found in table 86 that is not matched in table 76. When this happens, data is copied from mirror M1 to mirror M0 or from mirror M0 to mirror M1 according to the table entry that lacks the matching entry. To illustrate, tables 76 and 86 are not identical because table 76 includes an entry m consisting of a tag number 51 and logical block 30, and this entry cannot be found within table 86. To bring these mirrors M0 and M1 back into synchronization, data is copied from logical block 30 of mirror M0 to logical block 30 of mirror M1. Thereafter, entry m is deleted from tag table 76. Once this copying process is complete, and assuming there are no other mismatching entries between tables 76 and 86, mirrors M1 and M0 are back in synchronization.

[0035] In addition to creating or deleting entries in tag tables 76 and 86, tag table managers 72 and 82 can operate to (1) identify instances where mirrors M0 and M1 are out of synchronization or (2) delete matching entries in tag tables 76 and 86. Figure 6 illustrates these operational aspects of tag table manager 72. The operational aspects shown in Figure 6 can be initiated at any time. For example, the process shown in Figure 6 can be initiated after failure of host 42.

[0036] After the process is started in step 90, tag table manager 72 selects an entry in tag table 76 as shown in step 94 and then sends a message to tag table manager 82 asking if there is a corresponding match within tag table 86. The message sent to tag table manager 82 identifies the tag and logical block number or range of logical block numbers of the tag selected in step 92. Tag table manager 82 accesses tag table 86 using tag and logical block number or range of logical block numbers of the entry selected in step 92 that is provided in the message sent by tag table manager 72, and determines if a match exists as shown in step 94. If tag table manager 82 finds an entry in table 86 that matches the entry selected in table 76 (i.e., an entry in table 86 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in step 92), then tag table manager deletes the matching entry in table 86 and sends a reply message to tag table manager 72 indicating that table 86 did include a match to the entry selected in step 92. Tag manager 72, in response to receiving the message from tag table manager 82 that a matching entry did exist, deletes the entry selected in step 92. The entry deletion by tag managers 72 and 82 is shown in step 96. However, if tag table 86 does not have a match to the entry selected in table 76, then a message to that effect is sent back to tag table manager 72 and the process proceeds to step 100. In step 100, data is copied from mirror M0 to mirror M1. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M0 that are identified by the entry selected in step 92, is copied to mirror M1 at the logical memory block or range of logical memory blocks identified by the entry selected in step 92. For example, assuming the selected entry in step 92 is entry m of table 76, the data contents of logical block 30 of mirror M0 is copied to logical block 30 of mirror M1 in step 100. After copying data from mirror M0 to mirror M1in step 100, tag table manager 72 deletes the entry selected in table 76 as shown in step 104. Thereafter, as shown in step 104, tag table manager 72 checks tag table 76 to see if any tag entries remain. If an

entry remains, then it is selected in step 92, and steps 94-102 are repeated. If no additional entries remain in table 76 at step 104, the process ends.

[0037] It is noted that tag table manager 82 operates under a process similar to that shown in Figure 6. In particular, tag table manager 82 selects an entry in tag table 86 and sends a message to tag table manager 72 asking if there is a corresponding match within tag table 76. The message sent to tag table manager 72 identifies the tag and logical block number or range of logical block numbers of the tag selected in table 86. Tag table manager 72 accesses tag table 76 using tag and logical block number or range of logical block numbers of the selected entry of table 86 provided in the message sent by tag table manager 82, and determines if a match exists in table 76. If tag table manager 72 finds an entry in table 76 that matches the entry selected in table 86 (i.e., an entry in table 76 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in table 86), then tag table manager 72 deletes the matching entry in table 76 and sends a reply message to tag table manager 82 indicating that table 76 did include a match to the entry selected in table 86. Tag manager 82, in response to receiving the message from tag table manager 72 that a matching entry did exist in table 76, deletes the selected entry in table 86. However, if tag table 76 does not have a match to the entry selected in table 86, then a message to that effect is sent to tag table manager 82 and data is copied from mirror M1 to mirror M0. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M1 that are identified by the entry selected in table 86 is copied to the logical memory block or range of logical memory blocks identified by the entry selected in table 86 but contained in mirror M0. After copying data from mirror M1 to mirror M0, tag table manager 82 deletes the selected entry in table 86. Thereafter, tag table manager 82 checks tag table 86 to see if any tag entries remain. If an entry remains, then it is selected, and the foregoing steps are repeated. If no additional entries remain in table 86, the process ends.

[0038] If host 42 crashes after sending a write IO transaction to disk array 44 but before sending the same IO transaction to disk array 46, that will leave M0 and M1 in inconsistent state. Before host 42 is restarted, synchronization authority will generate and send a message to disk arrays 44 and 46 instructing them to bring M0 and M1 into a consistent state as described by the foregoing sections.

[0039] While a resynchronization of mirrors M0 and M1 occur after, for example a crash of host 42, it may be necessary to suspend host 42 from further generation of IO transactions to write data to mirrors M0 and M1 until mirrors M0 and M1 are brought back into a consistent state.

[0040] As noted above, storage subsystems may take form in OSDs. In this embodiment, tag tables, such as tag tables 76 and 86, my have to be modified to include an object number and information indicating whether the corresponding write operation is overwriting existing data of the storage object or whether the corresponding write operation extends the file length.

[0041] It was noted above that in an alternative embodiment, the tag generator may be placed in one of the disk arrays 44 or 46. In this alternative embodiment, host 42 could forward the IO transaction from application 52 to disk array 44 (for example) that contains the tag generator with an instruction for disk array 44 to generate first and second write IO transactions that contain a unique tag. The first IO transaction would be provided to the storage manager 62 while the second IO transaction is transmitted to storage manager 64 in disk array 46.

[0042] Although the present invention has been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein. On the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the scope of the invention as defined by the appended claims.

WE CLAIM:

1. A method comprising:

a computer system generating first and second write transactions;

wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;

the computer system transmitting the first and second write transactions to first and second storage devices, respectively.

2. The method of claim 1 further comprising:

the computer system generating third and fourth write transactions;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;

the computer system transmitting the third and fourth write transactions to the first and second storage devices, respectively.

3. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a logical block of a first storage object;

the second write transaction comprises data D to be written to a logical block of a second storage object.

4. The method of claim 3 further comprising:

the first storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory;

the second storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

5. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

6. The method of claim 5 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an identity of the range of logical blocks of the first storage object where data D is to be written, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an identity of the range of logical blocks in the second storage object where data D is to be written, wherein the second tag table is stored in second memory.

7. The method of claim 4 further comprising comparing the contents of one entry in the first tag table with the contents of entries in the second tag table to determine whether the second tag table includes an entry that matches the one entry.

8. The method of claim 7 further comprising copying data, associated with the logical block number identified by the one entry, from the first storage object to the logical block in the second storage object if the second table lacks an entry with contents matching the contents of the one entry

9. The method of claim 7 further comprising deleting the one entry in the first table if the second table contains an entry with contents that match the contents of the one entry.

10. The method of claim 9 further comprising deleting the entry in the second table with contents that match the contents of the one entry.

11. The method of claim 1 further comprising:

the computer system generating a write transaction to write data to a logical block of a data volume;

the computer system incrementing a counter in response to generating the write transaction;

the computer system generating the first and second tags, wherein each of the first and second tags relate to the first and second write transactions, respectively, wherein the first and second tags are generated in response to generation of the write transaction, and wherein the first and second tags are generated as a function of an output of the incremented counter.

12. The method of claim 1 wherein the first and second storage devices comprise first and second object storage devices.

13. The method of claim 1 wherein:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

14. The method of claim 13 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an indication that data D is to be stored in the extension of the first storage object, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an indication that data D is to be stored in the extension of the second storage object, wherein the second tag table is stored in second memory.

15. A method comprising:

a computer system generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

the computer system transmitting the transaction to a storage device.

16. The method of claim 1:

wherein the computer system generates the first and second write transactions in response to generation of a write transaction by a first application executing on the computer system, wherein the first and second tags are generated by a first tag generator;

the computer system generating third and fourth transactions in response to generation of a write transaction by a second application executing on the computer system;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction, wherein the third and fourth tags are generated by a second tag generator.

17. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

generating first and second write transactions;

wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;

transmitting the first and second write transactions directly or indirectly to first and second storage devices, respectively.

18. The computer readable medium of claim 17 wherein the method further comprises:

generating third and fourth write transactions;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;

transmitting the third and fourth write transactions directly or indirectly to the first and second storage devices, respectively.

19. The computer readable medium of claim 17 wherein:

the first write transaction comprises data D to be written to a logical block of a first storage object;

the second write transaction comprises data D to be written to a logical block of a second storage object.

20. The computer readable medium of claim 17 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

21. The computer readable medium of claim 17 wherein the first tag is identical to the second tag.

22. The computer readable medium of claim 17 wherein the method further comprises:

generating a write transaction to write data to a logical block of a data volume;

incrementing a counter in response to generating the write transaction;

generating the first and second tags, wherein each of the first and second tags relate to the first and second write transactions, respectively, wherein the first and second tags are generated in response to generation of the write transaction, and wherein the first and second tags are generated as a function of an output of the incremented counter,.

23. The computer readable medium of claim 17 the first and second storage devices comprise first and second object storage devices.

24. The computer readable medium of claim 17:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

25. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

transmitting the transaction to a storage device.

26. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

in response to receiving a first transaction comprising a first tag, storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory, wherein the first tag corresponds to a second tag of a second write transaction;

the second storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIV E VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

ABSTRACT OF THE DISCLOSURE

[0043] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

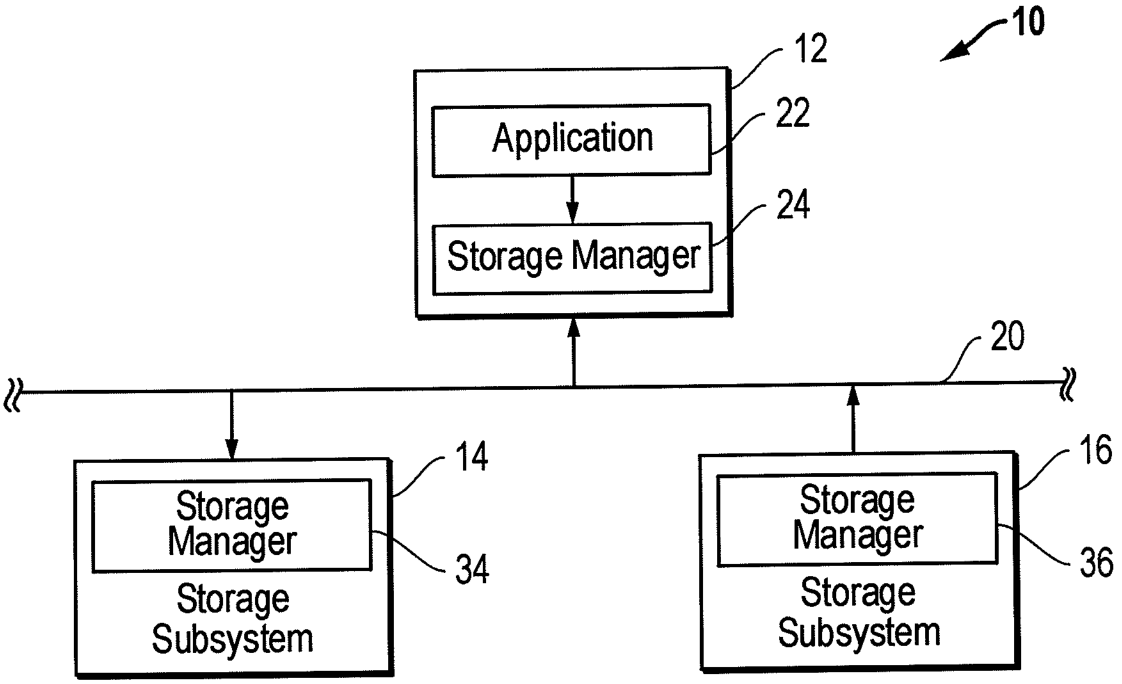

10
12
22
Application
24
Storage Manager
20
14
Storage Manager
34
Storage Subsystem
16
Storage Manager
36
Storage Subsystem

FIG. 1

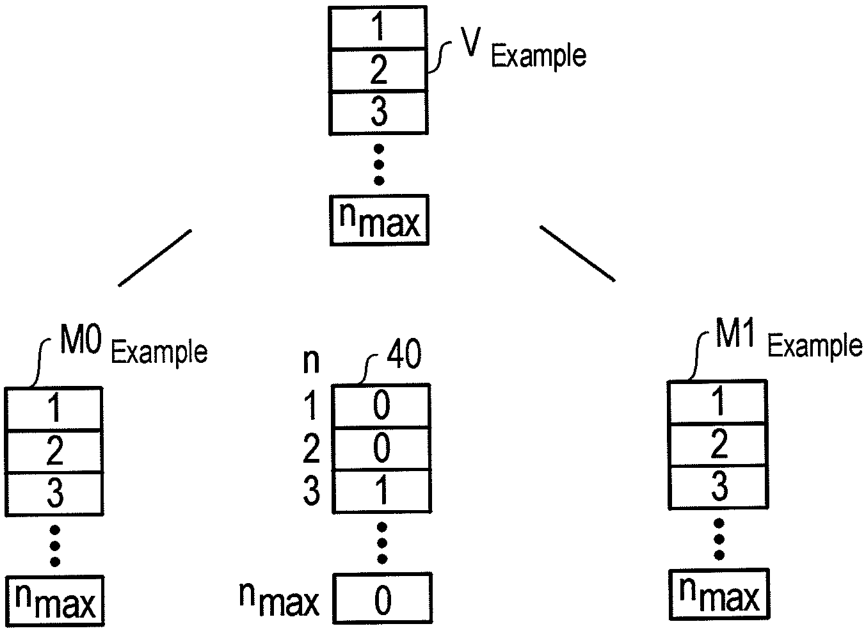

1
2
3
n_{max}
V Example
M0 Example
1
2
3
n_{max}
n
40
1 0
2 0
3 1
n_{max} 0
M1 Example
1
2
3
n_{max}

FIG. 2

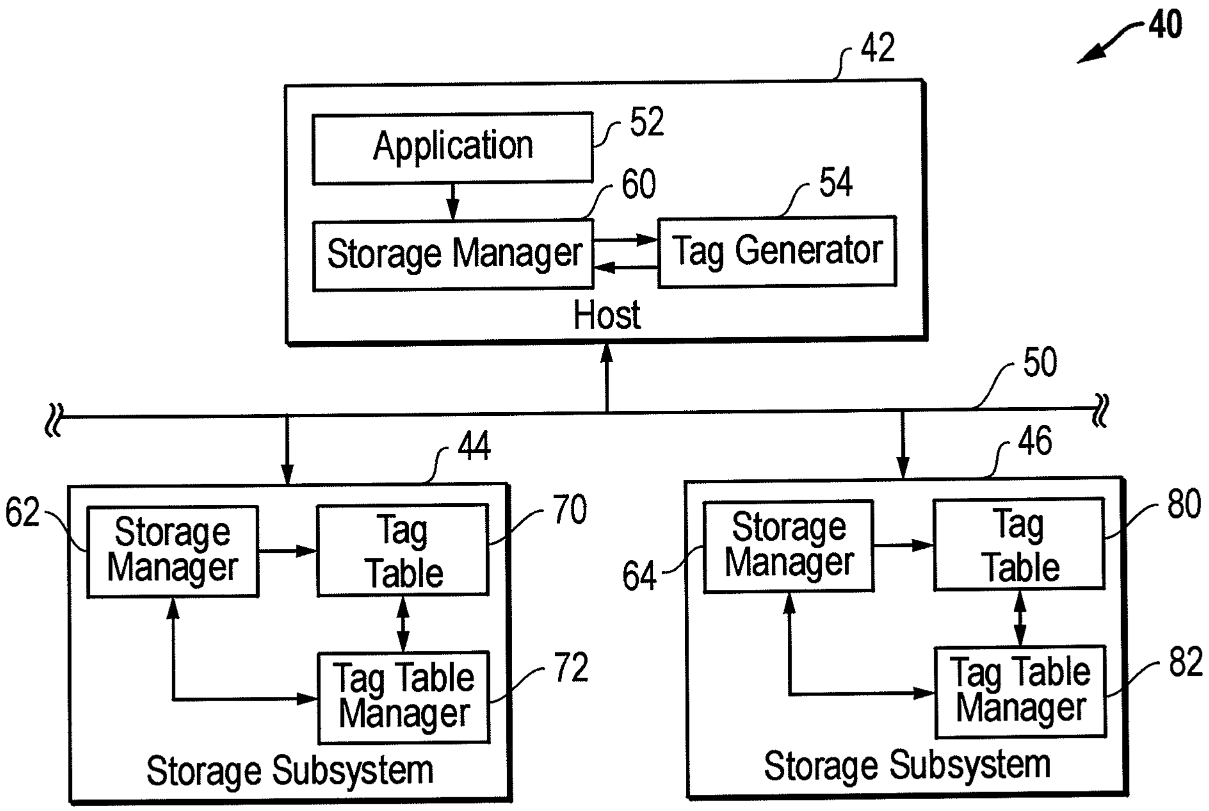

40
42
Application
52
60
54
Storage Manager
Tag Generator
Host
50
44
46
62
Storage Manager
Tag Table
70
64
Storage Manager
Tag Table
80
Tag Table Manager
72
Tag Table Manager
82
Storage Subsystem
Storage Subsystem

FIG. 3

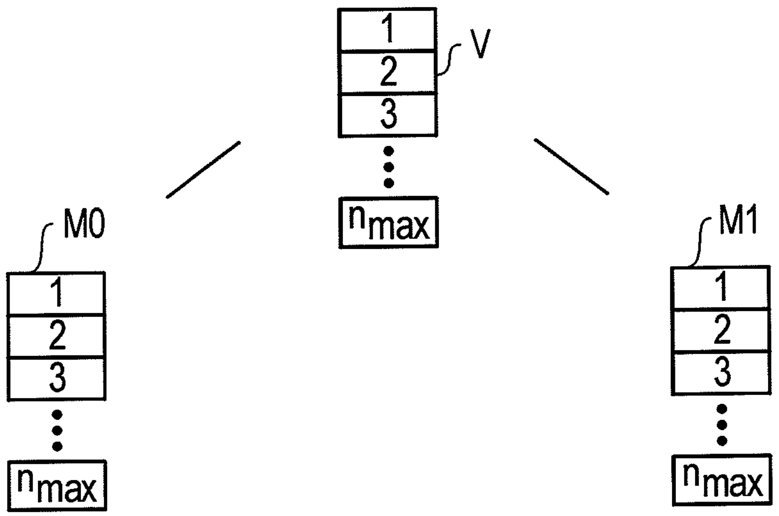
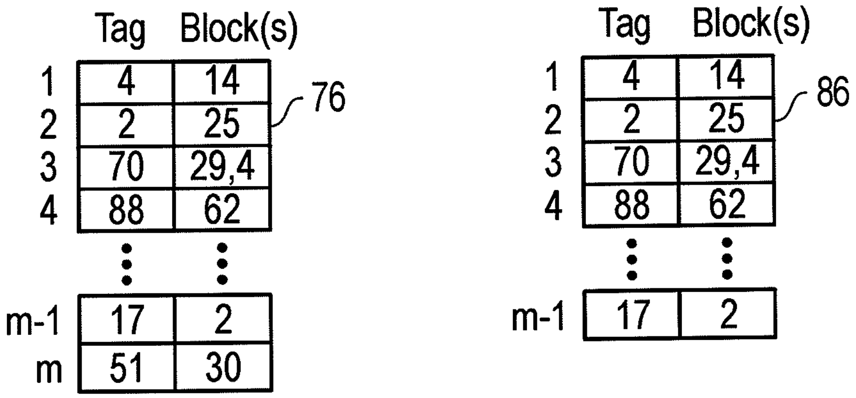

1
2
3
V
n_{max}
M0
1
2
3
n_{max}
M1
1
2
3
n_{max}

FIG. 4



	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
⋮	⋮	⋮
m-1	17	2
m	51	30

	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
⋮	⋮	⋮
m-1	17	2

FIG. 5

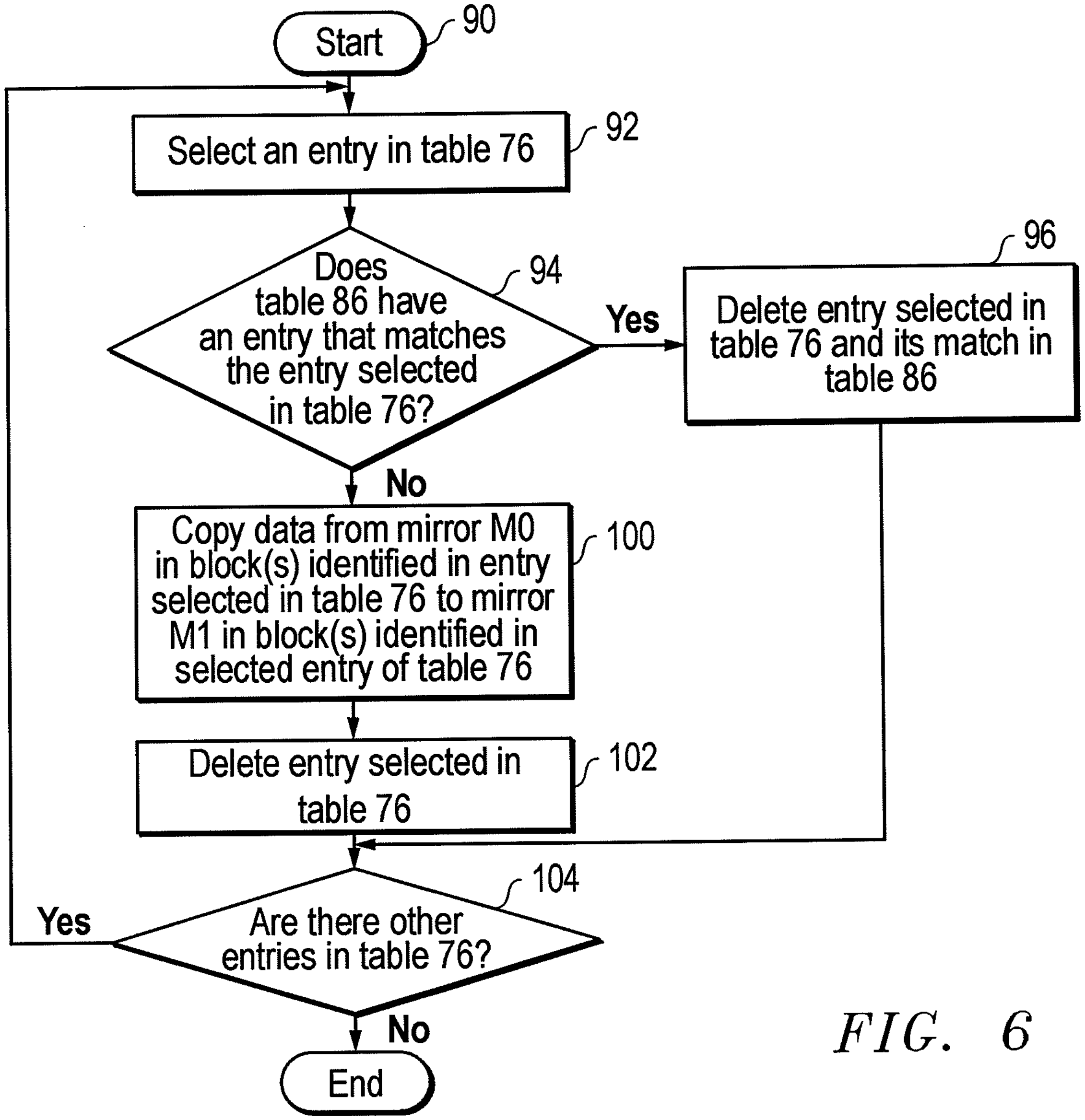

Start
90
Select an entry in table 76
92
Does table 86 have an entry that matches the entry selected in table 76?
94
Yes
Delete entry selected in table 76 and its match in table 86
96
No
Copy data from mirror M0 in block(s) identified in entry selected in table 76 to mirror M1 in block(s) identified in selected entry of table 76
100
Delete entry selected in table 76
102
Are there other entries in table 76?
104
Yes
No
End

FIG. 6

DECLARATION FOR PATENT APPLICATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below adjacent to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of subject matter (process, machine, manufacture, or composition of matter, or an improvement thereof) which is claimed and for which a patent is sought by way of the application entitled

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

which (check) ☒ is attached hereto.
□ and is amended by the Preliminary Amendment attached hereto.
□ was filed on ___ as Application Serial No. _____
□ and was amended on (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
Number	Country	Day/Month/Year Filed	Yes	No
			□	□

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

Provisional Application Number	Filing Date

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) or PCT international application(s) designating the United States of America listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal

Regulations, § 1.56, which became available between the filing date of the prior application(s) and the national or PCT international filing date of this application:

Application Serial No.	Filing Date	Status (patented, pending, abandoned)

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith:

Joseph FitzGerald (33,881); John Brigden (40,530); Julie Stephenson (41,330); and

Marc R. Ascolese (42,268); Brenna A. Brock (48,309); Sam G. Campbell (42,381); Justin M. Dillon (42,486); D'Ann Rifai (47,026); Eric A. Stephenson (38,321).

Please address all correspondence and telephone calls to:

Eric A. Stephenson
CAMPBELL STEPHENSON ASCOLESE LLP
4807 Spicewood Springs Road
Building 4, Suite 201
Austin, Texas 78759
Telephone: 512-439-5093
Facsimile: 512-439-5099

I declare that all statements made herein of my own knowledge are true, all statements made herein on information and belief are believed to be true, and all statements made herein are made with the knowledge that whoever, in any matter within the jurisdiction of the Patent and Trademark Office, knowingly and willfully falsifies, conceals, or covers up by any trick, scheme, or device a material fact, or makes any false, fictitious or fraudulent statements or representations, or makes or uses any false writing or document knowing the same to contain any false, fictitious or fraudulent statement or entry, shall be subject to the penalties including fine or imprisonment or both as set forth under 18 U.S.C. 1001, and that violations of this paragraph may jeopardize the validity of the application or this document, or the validity or enforceability of any patent, trademark registration, or certificate resulting therefrom.

Full name of sole (or first joint inventor): Ronald S. Karr

Inventor's Signature: ______________________ Date: ______________

Residence: Palo Alto, California 94301

Post Office Address: 333 Ramona Street
Palo Alto, California 94301

Citizenship: U.S.A.

Full name of second inventor: Ramana Jonnala
Inventor's Signature: ______________________ Date: ______________
Residence: Sunnyvale, California 94086
Post Office Address: 825 E. Evelyn Avenue, #326
Sunnyvale, California 94086
Citizenship: U.S.A.

Full name of third inventor: Narasimha R. Valiveti
Inventor's Signature: ______________________ Date: ______________
Residence: Sunnyvale, California 94085
Post Office Address: 395 Ano Nuevo Avenue, Apt #413
Sunnyvale, California 94085
Citizenship: India

Full name of fourth inventor: Dhanesh Joshi
Inventor's Signature: ______________________ Date: ______________
Residence: Santa Clara, California 95051
Post Office Address: 46, Cabot Avenue
Santa Clara, California 95051
Citizenship: India

EXHIBIT T

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:57 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Final Draft And Declaration
Attachments: VRT0120US-Patent Application.DOC; VRT0120US-Dwgs fr Mario 2-20-04.pdf; VRT0120US-Declaration.doc

---

From: Eric Stephenson
Sent: Thursday, February 26, 2004 5:53 PM
To: 'Dhanesh Joshi'; Narasimha Valiveti (narasimha.valiveti@veritas.com); Ramana Jonnala (ramana@veritas.com)
Cc: Ron Karr; julie.stephenson@veritas.com; Roz Donaldson; Anne Castle
Subject: VRTS 0655 (VRT0120US) Final Draft And Declaration

Greetings,

Attached please find the final draft of the above identified patent application. This draft includes changes requested by Dhanesh and Ron.

If all is in order, please print, sign and fax the attached declaration to me at the number below. Thereafter send the original to Julie Stephenson.

After the application is filed, we will send an assignment for your signatures.

Thanks,

Eric

Eric Stephenson
Campbell Stephenson Ascolese LLP
4807 Spicewood Springs Road
Suite 4-201
Austin, Texas 78759
(512) 439-5093 Direct
(512) 439-5099 Fax
estephenson@csapatent.com

> THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR THE PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED ABOVE. This message and the information contained herein is a confidential, attorney-client privileged communication. If you are not the intended recipient, you have received this document in error, and any review, dissemination, distribution, or copying of this message is strictly prohibited. If you are an unintended recipient, please notify the sender immediately and delete this message and any copies thereof. Thank you.

EXHIBIT U

EXPRESS MAIL LABEL NO.:

(EV 304738055 US)

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

BACKGROUND OF THE INVENTION

[0001] Large scale data processing systems typically include several data storage subsystems (e.g., disk arrays) each containing many physical storage devices (e.g., hard disks) for storing critical data. Data processing systems often employ storage management systems to aggregate these physical storage devices to create highly reliable data storage. There are many types of highly reliable storage. Mirrored volume storage is an example. Mirrored volume storage replicates data over two or more mirrors of equal size. A logical memory block n of a mirrored volume maps to the same logical memory block n of each mirror. In turn, each logical memory block n of the mirrors map directly or indirectly to one or more disk blocks of one or more physical devices. Mirrored volumes provide data redundancy. If an application is unable to access data of one, a duplicate of the data sought should be available in an alternate mirror.

[0002] Figure 1 illustrates relevant components of an exemplary data processing system 10 that employs a two-way mirrored volume. While the present invention will be described with reference to a two-way mirrored volume, the present invention should not be limited thereto. The present invention may find use with other types of redundant storage including, for example, a three-way mirrored storage volume. Data processing system 10 includes a host (e.g., server computer system) 12 coupled to data storage subsystems 14 and 16 via storage interconnect 20. For purposes of explanation, storage interconnect 20 will take form in a storage area network (SAN) it being understood that the term storage interconnect should

not be limited thereto. SAN 20 may include devices (e.g., switches, routers, hubs, etc.) that cooperate to transmit input/output (IO) transactions between host 12 and storage subsystems 14 and 16.

[0003] Each of the data storage subsystems 14 and 16 includes several physical storage devices. For purposes of explanation, data storage subsystems 14 and 16 are assumed to include several hard disks. The term physical storage device should not be limited to hard disks. Data storage subsystems 14 and 16 may take different forms. For example, data storage subsystem 14 may consist of "Just a Bunch of Disks" (JBOD) connected to an array controller card. Data storage subsystem 16 may consist of a block server appliance. For purposes of explanation, each of the data storage subsystems 14 and 16 will take form in an intelligent disk array, it being understood that the term data storage subsystem should not be limited thereto.

[0004] As noted, each of the disk arrays 14 and 16 includes several hard disks. The hard disk is the most popular permanent storage device currently used. A hard disk's total storage capacity is divided into many small chunks called physical memory blocks or disk blocks. For example, a 10 GB hard disk contains 20 million disk blocks, with each block able to hold 512 bytes of data. Any random disk block can be written to or read from in about the same time, without first having to read or write other disk blocks. Once written, a disk block continues to hold data even after the hard disk is powered down. While hard disks in general are reliable, they are subject to occasional failure. Data systems employ data redundancy schemes such as mirrored volumes to protect against occasional failure of a hard disk.

[0005] Host 12 includes an application 22 executing on one or more processors. Application 22 generates IO transactions to access critical data in response to receiving requests from client computer systems (not shown) coupled to host 12. In addition to application 22, host 12 includes a storage manager 24 executing on one or more processors. Volume Manager^{TM} provided by VERITAS Software Corporation of Mountain View, California, is an exemplary storage manager. Although many of the examples described herein will emphasize virtualization architecture and terminology associated with the VERITAS Volume Manager™, the software and techniques described herein can be used with a variety of different storage managers and architectures.

[0006] Storage managers can perform several functions. More particularly, storage managers can create storage objects (also known as virtualized disks) by aggregating hard disks such as those of disk arrays 14 and 16, underlying storage objects, or both. A storage object is an abstraction. Figure 2 shows a visual representation of exemplary storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ created for use in data processing system 10. Each of the storage objects $V_{Example}$, $M0_{Example}$, and $M1_{Example}$ in Figure 2 consists of an array of n_{max} logical memory blocks that store or are configured to store data. While it is said that a logical memory block stores or is configured to store data, in reality the data is stored in one or more disk blocks of hard disks allocated directly or indirectly to the logical memory block.

[0007] Storage objects aggregated from hard disks can themselves be aggregated to form storage objects called logical data volumes. Logical data volumes are typically presented for direct or indirect use by an application such as application 22 executing on host 12. Thus, application 22 generates IO transactions to read data from or write data to one or more logical memory blocks of a data volume not knowing that the data volume is an aggregation of underlying storage objects, which in turn are aggregations of hard disks. Properties of storage objects depend on how the underlying storage objects or hard disks are aggregated. In other words, the method of aggregation determines the storage object type. In theory, there are a large number of possible methods of aggregation. The more common forms of aggregation include concatenated storage, striped storage, RAID storage, or mirrored storage. A more thorough discussion of how storage objects or hard disks can be aggregated can be found within Dilip M. Ranade [2002], "Shared Data Clusters" Wiley Publishing, Inc., which is incorporated herein by reference in its entirety.

[0008] Mirrored volumes provide highly reliable access to critical data. $V_{Example}$ of Figure 2 is an exemplary two-way mirrored volume. $V_{Example}$ was created by aggregating underlying storage objects (hereinafter mirrors) $M0_{Example}$ and $M1_{Example}$. Mirrors $M0_{Example}$ and $M1_{Example}$ were created by concatenating disk blocks from hard disks $d0_{Example}$ and $d1_{Example}$ (not shown) in disk arrays 14 and 16, respectively.

[0009] Storage managers can create storage object descriptions that describe the relationship between storage objects and their underlying storage objects or hard disks. These storage object descriptions typically include configuration maps. It is noted that

storage object descriptions may include other information such as information indicating that a storage object is a snapshot copy of another storage object.

[0010] A configuration map maps a logical memory block of a corresponding storage object to one or more logical memory blocks of one or more underlying storage objects or to one or more disk blocks of one or more hard disks. To illustrate, configuration maps $CMV_{Example}$, $CMM0_{Example}$, and $CMM1_{Example}$ are created for mirrored volume $V_{Example}$ and underlying mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Configuration map $CMV_{Example}$ maps each logical memory block n of $V_{Example}$ to logical memory blocks n of mirrors $M0_{Example}$ and $M1_{Example}$. Configuration map $CMM0_{Example}$ maps each logical memory block n of mirror $M0_{Example}$ to a disk block x in hard disk $d0_{Example}$, while configuration map $CMM1_{Example}$ maps each logical memory block n of mirror $M1_{Example}$ to a disk block y in hard disk $d1_{Example}$. Configuration map $CMV_{Example}$ can be provided for use by storage manager 24, while configuration maps $CMM0_{Example}$ and $CMM1_{Example}$ can be provided for use by storage managers 34 and 36 executing on one or more processors in disk arrays 14 and 16, respectively.

[0011] Storage managers use configuration maps to translate IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks. To illustrate, presume an IO transaction is generated by application 22 to write data D to logical memory block 3 of data volume $V_{Example}$. This IO transaction is received directly or indirectly by storage manager 24. In turn, storage manager 24 accesses configuration map $CMV_{Example}$ and learns that logical memory block 3 is mapped to logical block 3 in both mirrors $M0_{Example}$ and $M1_{Example}$. It is noted storage manager may not receive the exact IO transaction generated by application 22. However, the transaction storage manager 24 receives will indicate that data D is to be written to block 3 of mirrored volume $V_{Example}$.

[0012] Storage manager 24 then generates first and second IO transactions to write data D to logical blocks 3 and mirrors $M0_{Example}$ and $M1_{Example}$, respectively. The IO transactions generated by storage manager 24 are transmitted to disk arrays 14 and 16, respectively, via SAN 20. Storage managers 34 and 36 of disk arrays 14 and 16, respectively receive directly or indirectly, the IO transactions sent by storage manager 24. It is noted that the IO

transactions received by storage managers 34 and 36 may not be the exact IO transactions generated and sent by storage manager 24. Nonetheless, storage managers 34 and 36 will each receive an IO transaction to write data D to logical block 3 in mirrors $M0_{Example}$ and $M1_{Example}$, respectively. Storage manager 34, in response to receiving the IO transaction, accesses configuration map $CMM0_{Example}$ to learn that block 3 of mirror $M0_{Example}$ is mapped to, for example, disk block 200 within hard disk d0. In response, a transaction is generated to write data D to disk block 200 within disk d0. Storage manager 36 accesses configuration map $CMM1_{Example}$ to learn that logical block 3 of mirror $M1_{Example}$ is mapped to, for example, disk block 300 within hard disk d1. In response, an IO transaction is generated for writing data D to disk block 300 within hard disk d1.

[0013] As noted above, while hard disks are reliable, hard disks are subject to failure. Data may be inaccessible within a failed hard disk. For this reason and others, administrators create mirrored data volumes. Unfortunately, data within mirrors of a mirrored volume may get into an inconsistent state if, for example, there is a server crash, storage power failure, or other problem which prevents data from being properly written to a hard disk. Consider the exemplary mirrored volume $V_{Example}$. Presume storage manager 24 generates first and second IO transactions to write data D to logical blocks 3 in mirrors $M0_{Example}$ and $M1_{Example}$ in response to the IO transaction generated by application 22. Further, presume host 12 may fail after the first IO transaction is transmitted to disk array 14, but before the second IO transaction is transmitted to disk array 16. As a result, data D is written to disk block 200 within disk d0 of disk array 14, but data D is not written to disk block 300 within hard disk d1. When host 12 is restarted and exemplary volume $V_{Example}$ is made available again to application 22, mirrors $MO_{Example}$ and $M1_{Example}$ are said to be out of sync. In other words, mirrors $M0_{Example}$ and $M1_{Example}$ are no longer identical since at least block 3 in each contain different data. An IO transaction to read from logical memory block 3 of mirrored volume $V_{Example}$ could return either old or new data depending on whether the data is read from disk block 200 of hard disk d0 or disk block 300 of hard disk d1. Mirrors $MO_{Example}$ and $M1_{Example}$ should be resychronized before either is accessed again.

[0014] A brute force method to resynchronize mirrors $M0_{Example}$ and $M1_{Example}$ is simply to presume that one mirror (e.g., $M0_{Example}$) contains correct data and copy the contents of the one mirror to the other (e.g., $M1_{Example}$). It can take hours to resynchronize using this method.

A smarter resynchronization technique is possible, but it requires some preparation. This alternate technique involves using what is called a dirty region map. Figure 2 illustrates a visual representation of an exemplary dirty region map 40 consisting of n_{max} entries corresponding to the n_{max} logical memory blocks within mirrors $M0_{Example}$ and $M1_{Example}$. Each entry of the dirty region map 40 indicates whether the corresponding blocks in mirrors are considered synchronized. For example, if entry n is set to logical 1, then blocks n in the mirrors are considered out of synchronization, and if n is set to logical 0, blocks n in the mirrors are considered in synchronization. Entry n in dirty region map 40 is set to logical 1 when the application 22 generates a transaction for writing data to logical block n of volume $V_{Example}$. Entry n is maintained in the logical 1 state until acknowledgement is received from both disk arrays 14 and 16 that data D has been written successfully to the disk blocks allocated to logical memory block n. However, if, using the example above, the first IO transaction to write data D to block 3 of mirror $M0_{Example}$ succeeds while the second IO transaction to write to block 3 of $M1_{Example}$ fails, then entry 3 and dirty region map 40 will be maintained as a logical 1 indicating that logical blocks 3 in the mirrors are out of synchronization. Mirrors $M0_{Example}$ and $M1_{Example}$ can be resynchronized by copying data from $M0_{Example}$ to mirror $M1_{Example}$, but for only logical memory blocks corresponding to dirty region map entries set to logical 1.

SUMMARY OF THE INVENTION

[0015] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

Fig. 1 illustrates a block diagram of a data system in which a logical data volume may be created;

Fig. 2 illustrates a visual representation of the storage objects created in the data system of Fig. 1;

Fig. 3, illustrates a block diagram of a data system capable of implementing one embodiment of the present invention;

Figs. 4 shows visual representations of a mirrored data volume employed in the data system of Fig. 3;

Fig. 5 is a visual representation of exemplary tag tables employed in the disk arrays of Fig. 3;

Fig.6 is a flow chart illustrating relevant aspects of managing the tag tables shown in Fig. 5.

[0017] The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION

[0018] Figure 3 illustrates in block diagram form, relevant components of an exemplary data processing system 40 capable of employing one embodiment of the present invention. Data processing system 40 includes a host (e.g., server computer system) 42 coupled to data storage subsystems 44 an 46 via storage interconnect 50. The present invention should not be limited to use in a data processing system consisting of only two storage subsystems. The present invention may be employed in a data processing system consisting of more than two storage subsystems.

[0019] Storage interconnect 50 takes form in a SAN consisting of several devices (e.g., switches, routers, etc.) that operate to transmit data between devices 42-46 including IO transactions to read or write data, it being understood that the term storage interconnect should not be limited thereto. SAN 50 can operate according to any one of a number of different protocols including Fibre Channel, Ethernet, etc. Each of the devices 42-46 includes one or more processors capable of processing data according to instructions of a software component such as a storage manager. As such, each of the devices 42-46 can be considered a computer system.

[0020] Data storage subsystems 44 and 46 may take different forms. For example, data storage subsystems 44 and 46 may take form in object storage devices (OSDs). Unless otherwise noted, each of the data storage subsystems 44 and 46 is implemented as a disk array, it being understood that the term data storage subsystem should not be limited thereto. Each of the disk arrays 44 and 46 includes several hard disks. Moreover, each of the disk arrays 44 and 46 may include a memory for storing a tag table that will be more fully described below.

[0021] Host 42 includes an application 52 executing on one or more processors. Application 52 may take one of many different forms. For example, application 52 may take form in a database management system (DBMS), customer relationship management software, etc. Application 52 generates IO transactions to read data from or write data to a data volume in response to receiving requests from client computer systems (not shown) coupled to host 42. Host 42 also includes a tag generator 54 executing on one or more

processors. It is noted that tag generator, in an alternative embodiment, could be placed in a different device of system 40. For example, tag generator 54 could be a part of one of the disk arrays 44 or 46. For purposes of explanation, it will be presumed that the tag generator is placed in host 42 as shown in Figure 4, unless otherwise noted. Tag generator 54 will be more fully described below.

[0022] Devices 42-46 include a storage manager 60-64, respectively, executing on one or more processors. Each of the storage managers 60-64 is capable of performing many functions. More particularly, each of the storage managers 60-64 is capable of translating IO transactions directed to one storage object into one or more IO transactions that access data of one or more underlying storage objects or hard disks of disk arrays 44 and 46. Storage managers 60-64 translate IO transactions according to configuration maps provided thereto. For purposes of illustration, it will be presumed that application 52 is provided with access to a mirrored data volume V having two mirrors, storage objects M0 and M1. It is noted that the present invention can be employed with respect to a data volume consisting of more than two mirrors.

[0023] A visual representation of volume V and mirrors M0 and M1 is found within Figure 4. As shown in Figure 4, each of V, M0, and M1 consist of an array of n_{max} logical memory blocks. It is noted that mirrors M0 and M1 could be formed from underlying storage objects (e.g., logical units or LUNs), or that mirrors M0 and M1 could be created as striped or RAID storage over hard disks or LUNs. It will be presumed, however, that M0 is formed by aggregating disk blocks contained within disk d0 (not shown) of disk array 44, and that mirror M1 is formed by aggregating disk blocks contained within disk d1 of disk array 46.

[0024] Storage managers 60-64 are provided with storage object descriptions for storage objects V, M0, and M1, respectively. Generally, the storage object descriptions define the relationships between storage objects and their underlying storage objects or hard disks. Moreover, the storage object descriptions may include other information. For example, the storage object descriptions for M0 and M1 indicate that M0 and M1 are mirrors or each other. As such, storage manager 62 knows that a mirror of storage object M0 is contained within storage subsystem 46, and storage manager 64 knows that a mirror of storage object M1 is contained within storage subsystem 44.

[0025] Each of the storage object descriptions may also include a configuration map. A configuration map CMV is provided to storage manager 60 which maps each logical block n of volume V to corresponding blocks n in mirrors M0 and M1. Storage manager 62 is provided with a configuration map CMM0 which maps each logical block n of mirror M0 to a disk block x within hard disk d0. Lastly, storage manager 64 is provided with a configuration map CMM1 which maps each logical block n of mirror M1 to a disk block y within hard disk d1. Storage manager 60 is capable of translating IO transactions received from application 52 using configuration map CMV. Thus, when application 52 generates a transaction to write a data to block n of volume V, storage manager in response generates first and second transactions for writing a data D to blocks n in mirrors M0 and M1, respectively. Storage manager 62, in response to receiving an IO transaction to write data to block n of mirror M0, uses its configuration map CMM0 to identify the corresponding disk block x within disk d0 to which data D is to be written. Likewise, storage manager 64, in response to receiving an IO transaction to write data D to block n of mirror M1, uses its configuration map CMM1 to identify the corresponding disk block y within disk d1 to which data D is to be written.

[0026] IO transactions are received by storage manager 60. Tag generator 54, in response to storage manager 60 receiving an IO transaction to write data, generates a tag unique to the IO transaction to write data. This tag is provided to storage manager 60. In one embodiment, the tag may be the count value of a counter. In this embodiment, the count value is incremented by one each time storage manager 60 receives an IO transaction to write data.

[0027] Storage manager 60 accesses its configuration map CMV in response to receiving an IO transaction to write data to logical memory block n of volume V or a range of logical memory blocks beginning with block n. For purposed of explanation, it will be presumed that storage manager 60 receives IO transactions to write data to one logical block n of volume V unless otherwise noted. From configuration map CMV, storage manager 60 learns that logical block n of volume V is mapped to a logical blocks n in mirrors M1 and M0. Accordingly, storage manager 60 generates first and second IO transactions to write data D to block n of mirrors M0 and M1. Each of these first and second IO transactions also includes the unique tag generated by tag generator 54. The first and second IO transactions, including

the unique tag, are transmitted by host 42 to disk arrays 44 and 46, respectively, via SAN 50. It is noted that the protocol employed with SAN 50 may have to be modified to accommodate IO transactions that include a unique tag.

[0028] Storage manager 62 is in data communication with nonvolatile memory 70. Likewise, storage manager 64 is in data communication with nonvolatile memory 80. Nonvolatile memories 70 and 80 are configured to store tag tables, such as tag tables 76 and 86, respectively, shown within Figure 5. Tag tables 76 and 86 are accessible by tag table managers 72 and 82, respectively. As will be more fully described below, tag table managers 72 and 82 perform several functions including the creation or deletion entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 may take form in software instructions executing on one or more processors of disk arrays 44 and 46, respectively.

[0029] Tag tables 76 and 86 are associated with mirrors M0 and M1, respectively. It is noted that additional tag tables like tag table 76 or 86 may be contained in memories 70 and 80, where each additional tag table is associated with a mirror of a mirrored data volume provided for access to application 52. However, for purposes of explanation, it will be presumed that memories 70 and 80 store only tag tables 76 and 86, respectively.

[0030] Each tag table 76 and 86 includes a plurality of entries. Each entry stores a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number. Alternatively, each tag table entry stores a tag and a physical block number or a range of disk block numbers beginning with a first disk block number. For purposes of explanation, it will be presumed that each entry of the tag tables 76 and 86 includes a tag and a logical memory block number or a range of logical memory block numbers beginning with a first logical memory block number unless otherwise noted. To illustrate, entry 2 of tag table 76 includes a tag equal to 2 and a logical block number equal to 25, and entry 3 of tag table 76 includes a tag equal to 70 and a range of 4 logical blocks beginning with logical memory block number 29.

[0031] Tag tables 76 and 86 track write IO transactions received by storage managers 62 and 64, respectively. Entries in tag tables 76 and 86 are created by tag managers 72 and 82, respectively, each time storage managers 62 and 64, respectively, receive IO transactions to

write data to mirrors M0 and M1, respectively. The contents (i.e., the tag and logical block number or range of logical block numbers) of the tag table entries are defined by the IO write transactions received by storage managers 62 and 64. To illustrate, presume storage manager 62 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M0. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 62 provides the tag and the logical block n or range of logical block numbers of the received IO transaction, to tag manager 72. Tag table manager 72, in response, creates a new entry in tag table 76. Tag table manager 72 then stores into the newly created entry of table 76 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 62 is written to the disk block or blocks in disk d0 allocated to, according to CMM0, the logical block n or range of logical blocks beginning with block n of mirror M0. It is noted that the foregoing process is also employed by tag table manager 82. More particularly, presume storage manager 64 receives an IO transaction to write data to logical memory block n or a range of logical memory blocks beginning with block n of mirror M1. This IO transaction includes a tag generated by tag generator 54 that is unique to the IO transaction. Storage manager 64 provides the tag and the logical block n or range of logical block numbers to receiving the IO transaction, to tag manager 82. Tag table manager 82, in response, creates a new entry in tag table 86. Tag table manager 82 then stores into the newly created entry of table 86 the received tag and logical block n or range of logical block numbers beginning with logical block n. After the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, or before the tag and logical block n or range of logical block numbers beginning with logical block n is stored in the newly created entry, data of the IO transaction received by storage manager 64 is written to the disk block or blocks in disk d1 allocated to, according to CMM1, the logical block n or range of logical blocks beginning with block n of mirror M1. It is noted that in another embodiment, host 42 includes multiple, independent applications each of which is capable of generating a transaction for writing data. In this embodiment, each application may have its own tag generator that generates tags unique to that tag generator. Further in this embodiment, the tag manager of each disk array is capable of distinguishing between tags generated by different tag generators.

[0032] Tag tables 76 and 86 can be used to determine whether mirrors M0 and M1 are in synchronization. If the data processing system 40 operates perfectly, e.g., there is no crash of host 42 or interruption of power to disk arrays 44 and 46, then all IO transactions to write data to mirrors M1 and M0 should reach disk arrays 46 and 44, respectively, and the contents of tag tables 76 and 86 should be identical. However, if due to, for example, crash of host 42 one of two separate but related IO transactions do not reach disk array 44 or 46, then the contents of tag tables 76 and 86 should differ and mirrors M0 and M1 will be out of synchronization. To illustrate, suppose host 42 generates first and second IO transactions for writing data D_{new} to logical block 30 in mirrors M0 and M1. Further, presume that each of these separate IO transactions includes a unique tag 51 generated by generator 54. Lastly, presume that disk array 44 receives the first IO transaction from host 42, but disk array 46 does not receive the second IO transaction. As a result, tag table manager 72 creates a new entry m in tag table 76 which includes tag 51 and block number 30 while no such corresponding tag entry is created in tag table 86. Tag table 76 indicates that logical block 30 of mirror M0 has been updated with data D_{new}. Since table 86 does not include a corresponding entry, it can be said that logical block 30 in mirror M1 has not been updated with data D_{new}, and mirrors M0 and M1 are out of synchronization.

[0033] In another embodiment, tokens can be generated by disk arrays 44 and 46 when IO transactions are received. To illustrate, presume storage managers 62 and 64 receive first and second IO transactions to write data to logical block n or a range of logical blocks beginning with block n of mirrors M0 and M1, respectively. Each of the first and second IO transactions include the same tag generated by tag generator 54 and which is unique to the first and second IO transactions. Storage manager 62 provides the tag and the logical block n or range of logical block numbers for the received first IO transaction, to tag manager 72. Likewise, storage manager 64 provides the tag and the logical block n or range of logical block numbers for the received second IO transaction, to tag manager 82. Tag table managers 72 and 82, in response, create new entries in tag tables 76 and 86, respectively. Tag table managers 72 and 82 then store into the newly created entries of tables 76 and 86 the tag and logical block n or range of logical block numbers beginning with logical block n. Thereafter each of tag managers 72 and 82 generate first and second tokens, respectively.

The first and second tokens can be used to identify the newly created entries in tables 76 and 86, respectively. The tokens can be simple numbers to identify corresponding entries in the tag tables. If and when storage managers 62 and 64 return status of the first and second IO transactions, respectively, to host 42, storage managers 62 and 64 may also return the first and second tokens associated with the newly generated tag entries in tables 76 and 86, respectively. The first and second tokens could be identical to each other, but they are not required to be identical. Host 42 receives status from both disk arrays 44 and 46, and host 42 declares the first and second IO transactions as complete. Host 42 will receive the first and second tokens. Presume now that host 42 generates third and fourth IO transactions to write data to block m of mirrors M0 and M1, respectively. Because host 42 received status reply for the first and second transactions, host 42 may add the first and second tokens to the third and fourth IO transactions before they are sent to disk arrays 44 and 46, respectively. Tag managers 72 and 82 are provided with the first and second tokens. Tag managers 72 and 82 delete entries in tables 76 and 86, respectively, that correspond to the first and second tokens, respectively, received from the host. In yet another alternative, host 42 can choose a certain time interval to send to disk arrays 44 and 46 a message consisting of several tokens, but only those tokens that are ready to be sent back to disk arrays 44 and 46. A token is ready to be sent back only after all related IO transactions are completed. The several tokens are subsequently provided to tag table managers 72 and 82, and tag managers 72 and 82 then delete entries corresponding to the several tokens.

[0034] When mirrors M0 and M1 are out of synchronization, table 76 and 86 can be used to bring mirrors M1 and M0 back into synchronization. Mirrors M0 and M1 will be out of synchronization when an entry is found within table 76 that is not matched in table 86, or when an entry is found in table 86 that is not matched in table 76. When this happens, data is copied from mirror M1 to mirror M0 or from mirror M0 to mirror M1 according to the table entry that lacks the matching entry. To illustrate, tables 76 and 86 are not identical because table 76 includes an entry m consisting of a tag number 51 and logical block 30, and this entry cannot be found within table 86. To bring these mirrors M0 and M1 back into synchronization, data is copied from logical block 30 of mirror M0 to logical block 30 of mirror M1. Thereafter, entry m is deleted from tag table 76. Once this copying process is complete, and assuming there are no other mismatching entries between tables 76 and 86, mirrors M1 and M0 are back in synchronization.

[0035] In addition to creating or deleting entries in tag tables 76 and 86, tag table managers 72 and 82 can operate to (1) identify instances where mirrors M0 and M1 are out of synchronization or (2) delete matching entries in tag tables 76 and 86. Figure 6 illustrates these operational aspects of tag table manager 72. The operational aspects shown in Figure 6 can be initiated at any time. For example, the process shown in Figure 6 can be initiated after failure of host 42.

[0036] After the process is started in step 90, tag table manager 72 selects an entry in tag table 76 as shown in step 94 and then sends a message to tag table manager 82 asking if there is a corresponding match within tag table 86. The message sent to tag table manager 82 identifies the tag and logical block number or range of logical block numbers of the tag selected in step 92. Tag table manager 82 accesses tag table 86 using tag and logical block number or range of logical block numbers of the entry selected in step 92 that is provided in the message sent by tag table manager 72, and determines if a match exists as shown in step 94. If tag table manager 82 finds an entry in table 86 that matches the entry selected in table 76 (i.e., an entry in table 86 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in step 92), then tag table manager deletes the matching entry in table 86 and sends a reply message to tag table manager 72 indicating that table 86 did include a match to the entry selected in step 92. Tag manager 72, in response to receiving the message from tag table manager 82 that a matching entry did exist, deletes the entry selected in step 92. The entry deletion by tag managers 72 and 82 is shown in step 96. However, if tag table 86 does not have a match to the entry selected in table 76, then a message to that effect is sent back to tag table manager 72 and the process proceeds to step 100. In step 100, data is copied from mirror M0 to mirror M1. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M0 that are identified by the entry selected in step 92, is copied to mirror M1 at the logical memory block or range of logical memory blocks identified by the entry selected in step 92. For example, assuming the selected entry in step 92 is entry m of table 76, the data contents of logical block 30 of mirror M0 is copied to logical block 30 of mirror M1 in step 100. After copying data from mirror M0 to mirror M1in step 100, tag table manager 72 deletes the entry selected in table 76 as shown in step 104. Thereafter, as shown in step 104, tag table manager 72 checks tag table 76 to see if any tag entries remain. If an

entry remains, then it is selected in step 92, and steps 94-102 are repeated. If no additional entries remain in table 76 at step 104, the process ends.

[0037] It is noted that tag table manager 82 operates under a process similar to that shown in Figure 6. In particular, tag table manager 82 selects an entry in tag table 86 and sends a message to tag table manager 72 asking if there is a corresponding match within tag table 76. The message sent to tag table manager 72 identifies the tag and logical block number or range of logical block numbers of the tag selected in table 86. Tag table manager 72 accesses tag table 76 using tag and logical block number or range of logical block numbers of the selected entry of table 86 provided in the message sent by tag table manager 82, and determines if a match exists in table 76. If tag table manager 72 finds an entry in table 76 that matches the entry selected in table 86 (i.e., an entry in table 76 has an identical tag and an identical logical block number or range of logical block numbers as that of the entry selected in table 86), then tag table manager 72 deletes the matching entry in table 76 and sends a reply message to tag table manager 82 indicating that table 76 did include a match to the entry selected in table 86. Tag manager 82, in response to receiving the message from tag table manager 72 that a matching entry did exist in table 76, deletes the selected entry in table 86. However, if tag table 76 does not have a match to the entry selected in table 86, then a message to that effect is sent to tag table manager 82 and data is copied from mirror M1 to mirror M0. More particularly, the data contents of the logical memory block or range of logical memory blocks in mirror M1 that are identified by the entry selected in table 86 is copied to the logical memory block or range of logical memory blocks identified by the entry selected in table 86 but contained in mirror M0. After copying data from mirror M1 to mirror M0, tag table manager 82 deletes the selected entry in table 86. Thereafter, tag table manager 82 checks tag table 86 to see if any tag entries remain. If an entry remains, then it is selected, and the foregoing steps are repeated. If no additional entries remain in table 86, the process ends.

[0038] If host 42 crashes after sending a write IO transaction to disk array 44 but before sending the same IO transaction to disk array 46, that will leave M0 and M1 in inconsistent state. Before host 42 is restarted, synchronization authority will generate and send a message to disk arrays 44 and 46 instructing them to bring M0 and M1 into a consistent state as described by the foregoing sections.

[0039] While a resynchronization of mirrors M0 and M1 occur after, for example a crash of host 42, it may be necessary to suspend host 42 from further generation of IO transactions to write data to mirrors M0 and M1 until mirrors M0 and M1 are brought back into a consistent state.

[0040] As noted above, storage subsystems may take form in OSDs. In this embodiment, tag tables, such as tag tables 76 and 86, my have to be modified to include an object number and information indicating whether the corresponding write operation is overwriting existing data of the storage object or whether the corresponding write operation extends the file length.

[0041] It was noted above that in an alternative embodiment, the tag generator may be placed in one of the disk arrays 44 or 46. In this alternative embodiment, host 42 could forward the IO transaction from application 52 to disk array 44 (for example) that contains the tag generator with an instruction for disk array 44 to generate first and second write IO transactions that contain a unique tag. The first IO transaction would be provided to the storage manager 62 while the second IO transaction is transmitted to storage manager 64 in disk array 46.

[0042] Although the present invention has been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein. On the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the scope of the invention as defined by the appended claims.

WE CLAIM:

1. A method comprising:

a computer system generating first and second write transactions;

wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;

the computer system transmitting the first and second write transactions to first and second storage devices, respectively.

2. The method of claim 1 further comprising:

the computer system generating third and fourth write transactions;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;

the computer system transmitting the third and fourth write transactions to the first and second storage devices, respectively.

3. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a logical block of a first storage object;

the second write transaction comprises data D to be written to a logical block of a second storage object.

4. The method of claim 3 further comprising:

the first storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory;

the second storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

5. The method of claim 1 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

6. The method of claim 5 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an identity of the range of logical blocks of the first storage object where data D is to be written, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an identity of the range of logical blocks in the second storage object where data D is to be written, wherein the second tag table is stored in second memory.

7. The method of claim 4 further comprising comparing the contents of one entry in the first tag table with the contents of entries in the second tag table to determine whether the second tag table includes an entry that matches the one entry.

8. The method of claim 7 further comprising copying data, associated with the logical block number identified by the one entry, from the first storage object to the logical block in the second storage object if the second table lacks an entry with contents matching the contents of the one entry

9. The method of claim 7 further comprising deleting the one entry in the first table if the second table contains an entry with contents that match the contents of the one entry.

10. The method of claim 9 further comprising deleting the entry in the second table with contents that match the contents of the one entry.

11. The method of claim 1 further comprising:

the computer system generating a write transaction to write data to a logical block of a data volume;

the computer system incrementing a counter in response to generating the write transaction;

the computer system generating the first and second tags, wherein each of the first and second tags relate to the first and second write transactions, respectively, wherein the first and second tags are generated in response to generation of the write transaction, and wherein the first and second tags are generated as a function of an output of the incremented counter.

12. The method of claim 1 wherein the first and second storage devices comprise first and second object storage devices.

13. The method of claim 1 wherein:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

14. The method of claim 13 further comprising:

the first object storage device receiving the first write transaction;

the first storage device storing in an entry of a first tag table, the first tag, an identity of the first storage object, and an indication that data D is to be stored in the extension of the first storage object, wherein the first tag table is stored in first memory;

the second object storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag, an identity of the second storage object, and an indication that data D is to be stored in the extension of the second storage object, wherein the second tag table is stored in second memory.

15. A method comprising:

a computer system generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

the computer system transmitting the transaction to a storage device.

16. The method of claim 1:

wherein the computer system generates the first and second write transactions in response to generation of a write transaction by a first application executing on the computer system, wherein the first and second tags are generated by a first tag generator;

a second computer system generating third and fourth transactions in response to generation of a write transaction by a second application executing on the second computer system;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction, wherein the third and fourth tags are generated by a second tag generator.

17. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

generating first and second write transactions;

wherein the first and second write transactions comprise first and second tags, respectively, wherein each of the first and second tags relate the first write transaction to the second write transaction;

transmitting the first and second write transactions directly or indirectly to first and second storage devices, respectively.

18. The computer readable medium of claim 17 wherein the method further comprises:

generating third and fourth write transactions;

wherein the third and fourth write transactions comprise third and fourth tags, respectively, wherein each of the third and fourth tags relate the third write transaction to the fourth write transaction;

transmitting the third and fourth write transactions directly or indirectly to the first and second storage devices, respectively.

19. The computer readable medium of claim 17 wherein:

the first write transaction comprises data D to be written to a logical block of a first storage object;

the second write transaction comprises data D to be written to a logical block of a second storage object.

20. The computer readable medium of claim 17 wherein:

the first write transaction comprises data D to be written to a range of logical blocks of a first storage object;

the second write transaction comprises data D to be written to a range of logical blocks of a second storage object.

21. The computer readable medium of claim 17 wherein the first tag is identical to the second tag.

22. The computer readable medium of claim 17 wherein the method further comprises:

generating a write transaction to write data to a logical block of a data volume;

incrementing a counter in response to generating the write transaction;

generating the first and second tags, wherein each of the first and second tags relate to the first and second write transactions, respectively, wherein the first and second tags are generated in response to generation of the write transaction, and wherein the first and second tags are generated as a function of an output of the incremented counter,.

23. The computer readable medium of claim 17 the first and second storage devices comprise first and second object storage devices.

24. The computer readable medium of claim 17:

the first write transaction comprises data D to be written to an extension of a first storage object;

the second write transaction comprises data D to be written to an extension of a second storage object.

25. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

generating a write transaction, wherein the write transaction comprises data to be written to a storage object and a tag unique to the write transaction;

transmitting the transaction to a storage device.

26. A computer readable medium storing instructions executable by a computer system, wherein the computer system implements a method in response to executing the instructions, the method comprising:

in response to receiving a first transaction comprising a first tag, storing in an entry of a first tag table, the first tag and an identity of the logical block where data D is to be written, wherein the first tag table is stored in first memory, wherein the first tag corresponds to a second tag of a second write transaction;

the second storage device receiving the second write transaction;

the second storage device storing in an entry of a second tag table, the second tag and an identity of the logical block where data D is to be written, wherein the second tag table is stored in second memory.

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIV E VIRTUAL STORAGE

Ronald S. Karr
Ramana Jonnala
Narasimha R. Valiveti
Dhanesh Joshi

ABSTRACT OF THE DISCLOSURE

[0043] Disclosed is a method implementable by a computer system for maintaining consistency between mirrors of a mirrored data volume. In one embodiment, the method includes the computer system generating first and second write transactions in response to the generation of transaction to write data to a mirrored data volume. The first and second write transactions comprise first and second tags, respectively. The first and second tags relate the first write transaction to the second write transaction. In one embodiment, the first and second tags are identical. After the first and second write transactions are generated, the computer system transmits the first and second write transactions to first and second storage subsystems, respectively. In one embodiment, the first and second storage subsystems store or are configured to store respective mirrors of the data volume. Additionally, each of the first and second storage subsystems include a tag table that stores tags contained in write transactions generated by the computer system. The tag tables can be used to track write transactions received by the first and second storage subsystems.

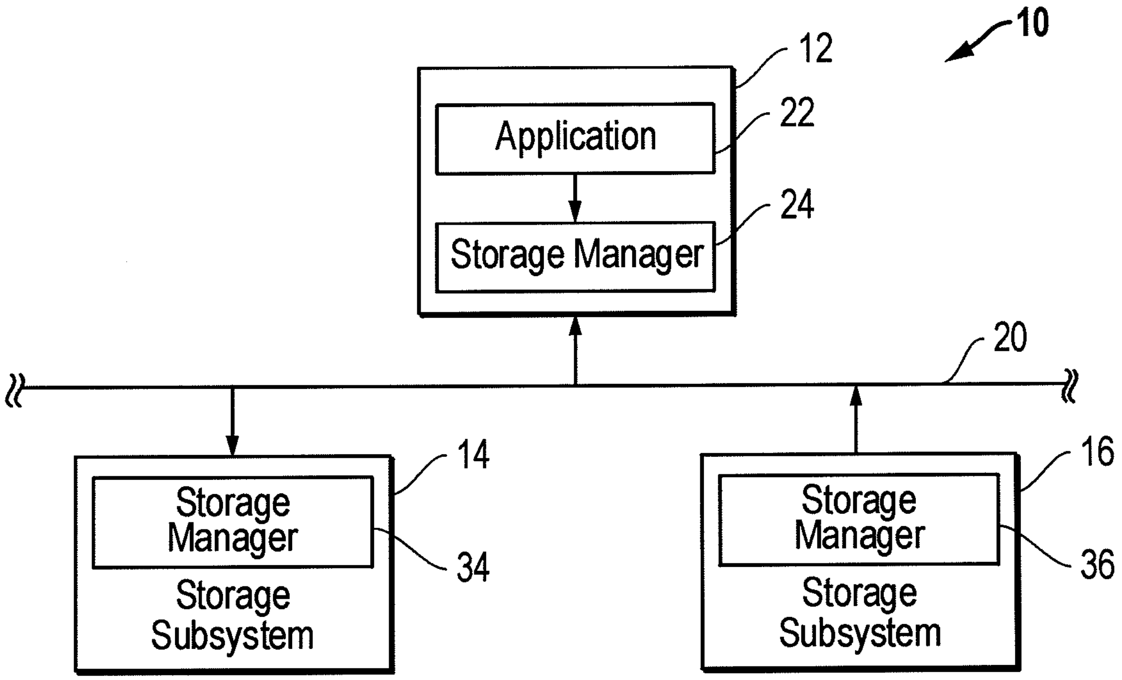

10
12
Application
22
Storage Manager
24
20
Storage Manager
Storage Subsystem
14
34
Storage Manager
Storage Subsystem
16
36

FIG. 1

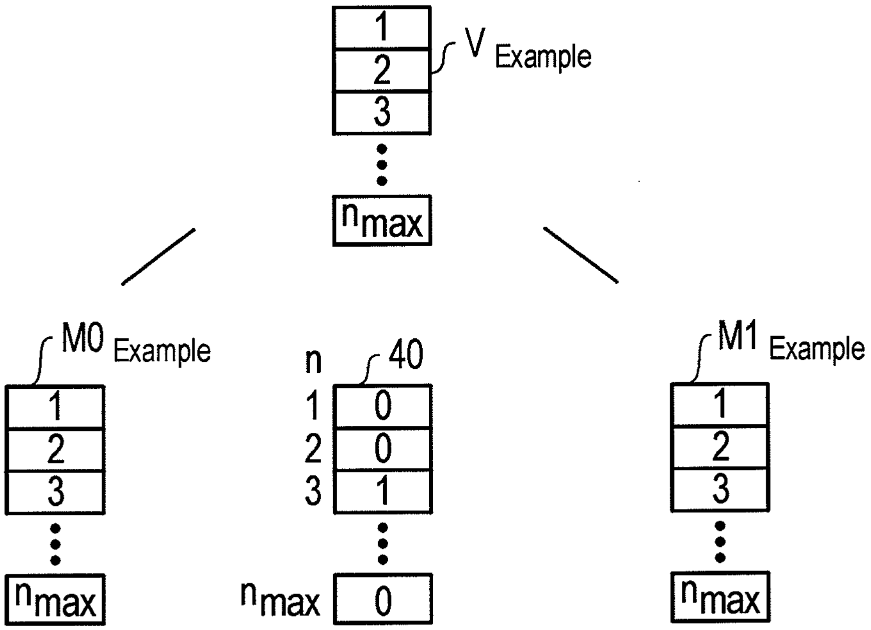

1
2
3
n_{max}
V Example
M0 Example
1
2
3
n_{max}
n
40
1 0
2 0
3 1
n_{max} 0
M1 Example
1
2
3
n_{max}

FIG. 2

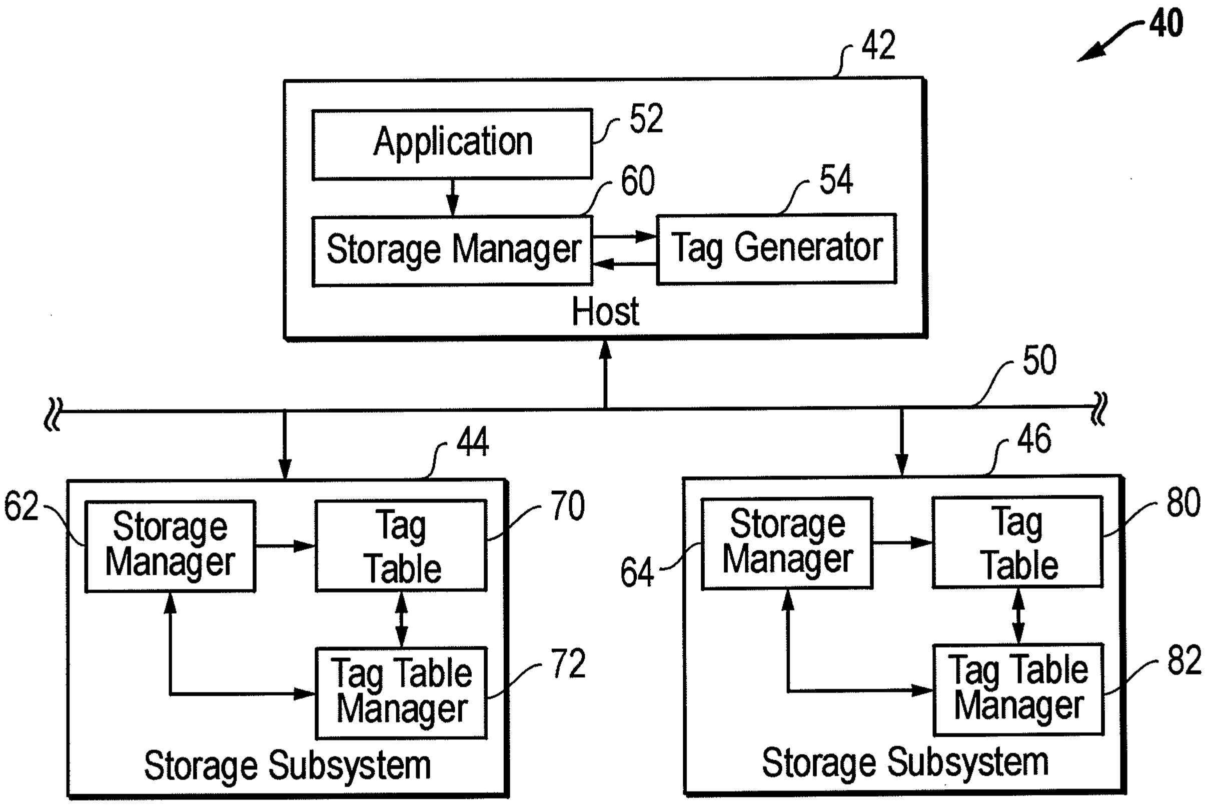

40
42
Application
52
60
54
Storage Manager
Tag Generator
Host
50
44
46
62
Storage Manager
Tag Table
70
Tag Table Manager
72
Storage Subsystem
64
Storage Manager
Tag Table
80
Tag Table Manager
82
Storage Subsystem

FIG. 3

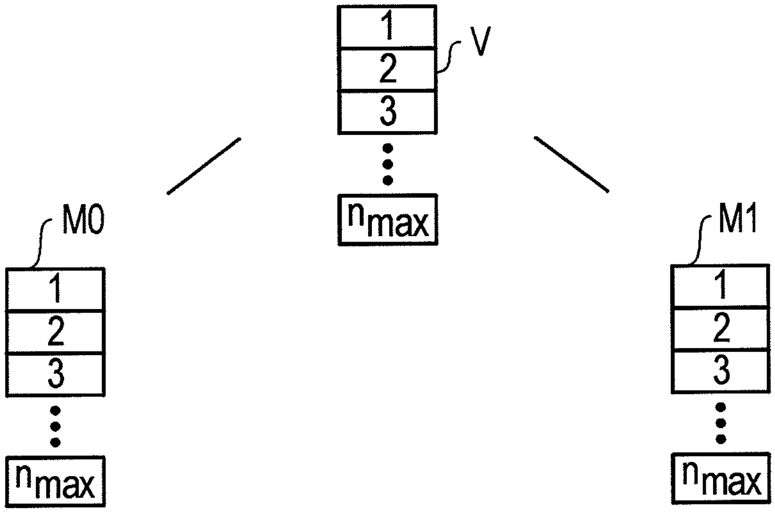

1
2
3
n_{max}
V
M0
1
2
3
n_{max}
M1
1
2
3
n_{max}

FIG. 4

	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
⋮	⋮	⋮
m-1	17	2
m	51	30

76

	Tag	Block(s)
1	4	14
2	2	25
3	70	29,4
4	88	62
⋮	⋮	⋮
m-1	17	2

86

FIG. 5

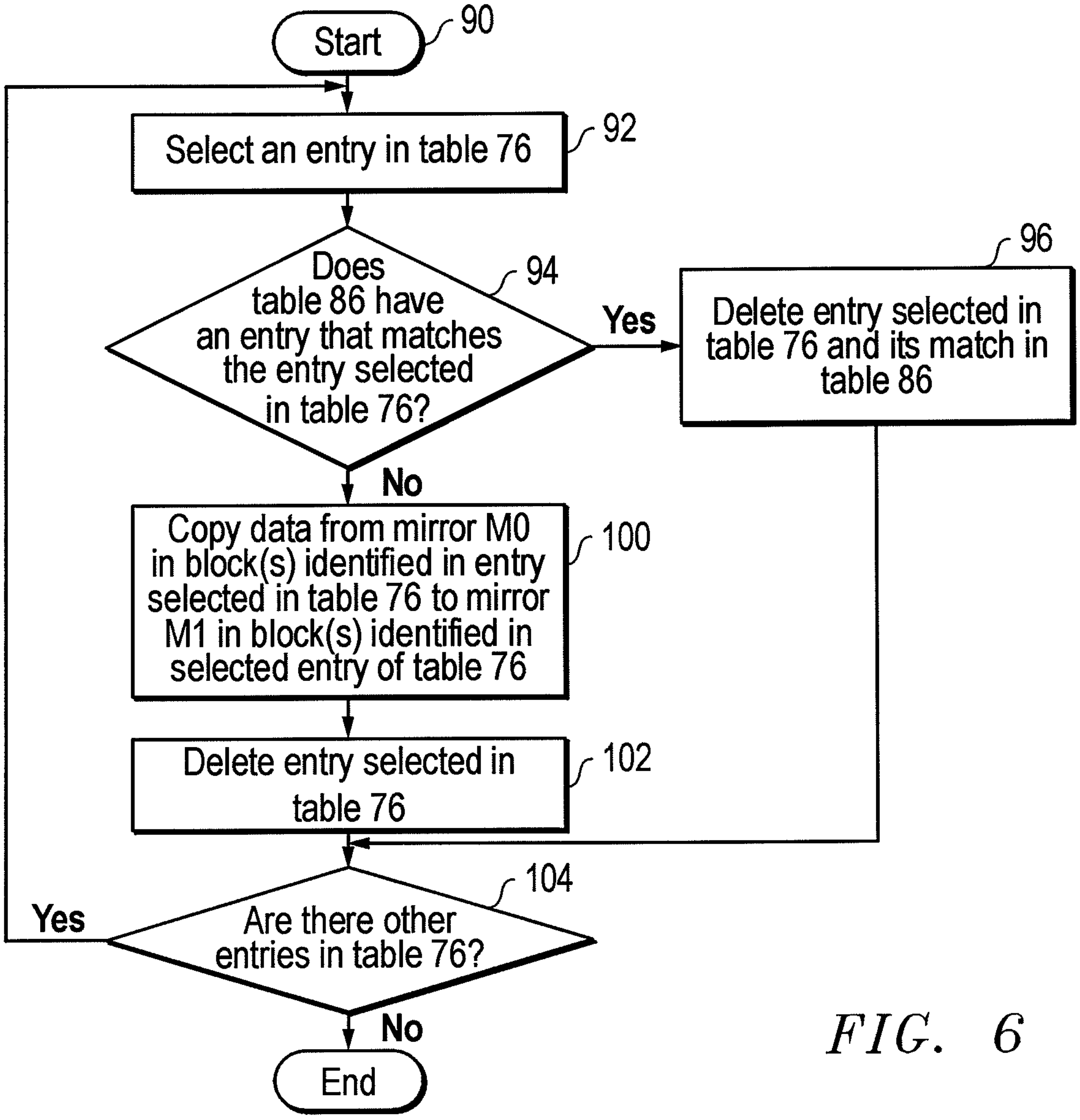

Start
90
Select an entry in table 76
92
Does table 86 have an entry that matches the entry selected in table 76?
94
Yes
Delete entry selected in table 76 and its match in table 86
96
No
Copy data from mirror M0 in block(s) identified in entry selected in table 76 to mirror M1 in block(s) identified in selected entry of table 76
100
Delete entry selected in table 76
102
Are there other entries in table 76?
104
Yes
No
End

FIG. 6

DECLARATION FOR PATENT APPLICATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below adjacent to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of subject matter (process, machine, manufacture, or composition of matter, or an improvement thereof) which is claimed and for which a patent is sought by way of the application entitled

TECHNIQUE FOR RECOVERING MIRROR CONSISTENCY IN COOPERATIVE VIRTUAL STORAGE

which (check) ☒ is attached hereto.
□ and is amended by the Preliminary Amendment attached hereto.
□ was filed on ___ as Application Serial No. ____
□ and was amended on (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
Number	Country	Day/Month/Year Filed	Yes	No
			□	□

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

Provisional Application Number	Filing Date

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) or PCT international application(s) designating the United States of America listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal

Regulations, § 1.56, which became available between the filing date of the prior application(s) and the national or PCT international filing date of this application:

Application Serial No.	Filing Date	Status (patented, pending, abandoned)

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith:

Joseph FitzGerald (33,881); John Brigden (40,530); Julie Stephenson (41,330); and

Marc R. Ascolese (42,268); Brenna A. Brock (48,309); Sam G. Campbell (42,381); Justin M. Dillon (42,486); D'Ann Rifai (47,026); Eric A. Stephenson (38,321).

Please address all correspondence and telephone calls to:

Eric A. Stephenson
CAMPBELL STEPHENSON ASCOLESE LLP
4807 Spicewood Springs Road
Building 4, Suite 201
Austin, Texas 78759
Telephone: 512-439-5093
Facsimile: 512-439-5099

I declare that all statements made herein of my own knowledge are true, all statements made herein on information and belief are believed to be true, and all statements made herein are made with the knowledge that whoever, in any matter within the jurisdiction of the Patent and Trademark Office, knowingly and willfully falsifies, conceals, or covers up by any trick, scheme, or device a material fact, or makes any false, fictitious or fraudulent statements or representations, or makes or uses any false writing or document knowing the same to contain any false, fictitious or fraudulent statement or entry, shall be subject to the penalties including fine or imprisonment or both as set forth under 18 U.S.C. 1001, and that violations of this paragraph may jeopardize the validity of the application or this document, or the validity or enforceability of any patent, trademark registration, or certificate resulting therefrom.

Full name of sole (or first joint inventor): Ronald S. Karr

Inventor's Signature: ________________ Date: __________

Residence: Palo Alto, California 94301

Post Office Address: 333 Ramona Street
Palo Alto, California 94301

Citizenship: U.S.A.

Full name of second inventor: Ramana Jonnala
Inventor's Signature: ______________________ Date: ______________
Residence: Sunnyvale, California 94086
Post Office Address: 825 E. Evelyn Avenue, #326
Sunnyvale, California 94086
Citizenship: U.S.A.

Full name of third inventor: Narasimha R. Valiveti
Inventor's Signature: ______________________ Date: ______________
Residence: Sunnyvale, California 94085
Post Office Address: 395 Ano Nuevo Avenue, Apt #413
Sunnyvale, California 94085
Citizenship: India

Full name of fourth inventor: Dhanesh Joshi
Inventor's Signature: ______________________ Date: ______________
Residence: Santa Clara, California 95051
Post Office Address: 46, Cabot Avenue
Santa Clara, California 95051
Citizenship: India

EXHIBIT V

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:58 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Final Draft And Declaration

Attachments: VRT0120US-Patent Application.DOC; VRT0120US-Dwgs fr Mario 2-20-04.pdf; VRT0120US-Declaration.doc

VRT0120US-Patent Application.D... VRT0120US-Dwgs fr Mario 2-20-0... VRT0120US-Declaration.doc (61 ...

```
-----Original Message-----
From: Eric Stephenson
Sent: Friday, February 27, 2004 10:45 AM
To: 'Ronald S. Karr'
Cc: Dhanesh Joshi; narasimha.valiveti@veritas.com; ramana@veritas.com; Ron Karr;
julie.stephenson@veritas.com; Roz Donaldson; Anne Castle
Subject: RE: VRTS 0655 (VRT0120US) Final Draft And Declaration
```

Greetings,

Attached is the final draft of the application with claim 16 revised according to Ron's comment below (thank you Ron).

Please print, sign, and fax the attached declaration to me at the fax number below. Thereafter, please forward the original signed declaration to Julie Stephenson.

Thanks,

Eric

```
-----Original Message-----
From: Ronald S. Karr [mailto:tron@veritas.com]
Sent: Thursday, February 26, 2004 6:55 PM
To: Eric Stephenson
Cc: Dhanesh Joshi; narasimha.valiveti@veritas.com; ramana@veritas.com; Ron Karr;
julie.stephenson@veritas.com; Roz Donaldson; Anne Castle
Subject: Re: VRTS 0655 (VRT0120US) Final Draft And Declaration
```

I looked at claim 16. It talks about two applications running on the same computer. My intention was to describe two independent computers that could use separate tag generators, not two applications on the same computer.

```
--
        Ronald S. Karr
        tron |-<=>-|    tron@veritas.com
```

On February 26, 2004, Eric Stephenson wrote:

```
 >Greetings,
 >
 >Attached please find the final draft of the above identified patent  >application.  This
draft includes changes requested by Dhanesh and Ron.  >
 >If all is in order, please print, sign and fax the attached declaration  >to me at the
number below. Thereafter send the original to Julie  >Stephenson.  >
 >After the application is filed, we will send an assignment for your  >signatures.  >
 >Thanks,
 >
 >Eric
```

>
>Eric Stephenson
>Campbell Stephenson Ascolese LLP
>4807 Spicewood Springs Road
>Suite 4-201
>Austin, Texas 78759
>(512) 439-5093 Direct
>(512) 439-5099 Fax
>estephenson@csapatent.com <mailto:estephenson@csapatent.com>
>
>
>
>THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR THE >PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED >ABOVE. This message and the information contained herein is a >confidential, attorney-client privileged communication. If you are not >the intended recipient, you have received this document in error, and >any review, dissemination, distribution, or copying of this message is > strictly prohibited. If you are an unintended recipient, please notify >the sender immediately and delete this message and any copies thereof. >Thank you.

Ronald Liu

From: Eric Stephenson
Sent: Monday, August 10, 2009 2:58 PM
To: Ronald Liu
Subject: FW: VRTS 0655 (VRT0120US) Final Draft And Declaration

-----Original Message-----
From: Ronald S. Karr [mailto:tron@veritas.com]
Sent: Thursday, February 26, 2004 6:55 PM
To: Eric Stephenson
Cc: Dhanesh Joshi; narasimha.valiveti@veritas.com; ramana@veritas.com; Ron Karr; julie.stephenson@veritas.com; Roz Donaldson; Anne Castle
Subject: Re: VRTS 0655 (VRT0120US) Final Draft And Declaration

I looked at claim 16. It talks about two applications running on the same computer. My intention was to describe two independent computers that could use separate tag generators, not two applications on the same computer.

--
Ronald S. Karr
tron |-<=>-| tron@veritas.com

On February 26, 2004, Eric Stephenson wrote:
>Greetings,
>
>Attached please find the final draft of the above identified patent >application. This draft includes changes requested by Dhanesh and Ron.
>
>If all is in order, please print, sign and fax the attached declaration >to me at the number below. Thereafter send the original to Julie >Stephenson.
>
>After the application is filed, we will send an assignment for your >signatures.
>
>Thanks,
>
>Eric
>
>Eric Stephenson
>Campbell Stephenson Ascolese LLP
>4807 Spicewood Springs Road
>Suite 4-201
>Austin, Texas 78759
>(512) 439-5093 Direct
>(512) 439-5099 Fax
>estephenson@csapatent.com <mailto:estephenson@csapatent.com>
>
>
>
>THE INFORMATION CONTAINED IN THIS MESSAGE IS INTENDED ONLY FOR THE >PERSONAL AND CONFIDENTIAL USE OF THE DESIGNATED RECIPIENT(S) NAMED >ABOVE. This message and the information contained herein is a >confidential, attorney-client privileged communication. If you are not >the intended recipient, you have received this document in error, and >any review, dissemination, distribution, or copying of this message is > strictly prohibited. If you are an unintended recipient, please notify >the sender immediately and delete this message and any copies thereof.
>Thank you.